National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

Computed Order Tracking

Applied to

Vibration Analysis of Rotating Machinery

BY

Erik Donald Stampe Munck

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

DEPARTMENT OF MECHANICAL ENGINEERING

Edmonton, Alberta

SPRING 1994

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-11377-9

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Erik Donald Stampe Munck

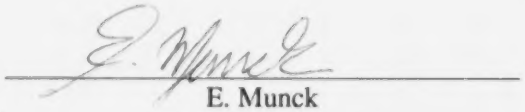
TITLE OF THESIS: Computed Order Tracking Applied to Vibration Analysis of Rotating Machinery

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1994

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



E. Munck
10128 111 Avenue
Grande Prairie, Alberta
T8V 1T6

15 April 1994

UNIVERSITY OF ALBERTA

FACULTY OF
GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *Computed Order Tracking Applied to Vibration Analysis of Rotating Machinery* submitted by Erik Donald Stampe Munck in partial fulfillment of the requirements for the degree of Master of Science.

K. R. Fyfe
Supervisor

A. Craggs

R. E. Rink

15 April 1994

On the possible future role of artificial mathematical constructs:

'So you see,' said Slartibartfast, slowly stirring his artificially constructed coffee, and thereby also stirring the whirlpool interfaces between real and unreal numbers, between the interactive perceptions of mind and Universe, and thus generating the restructured matrices of implicitly enfolded subjectivity which allowed his ship to reshape the very concept of time and space. 'how it is.'

'Yes,' said Arthur.

'Yes,' said Ford.

'What do I do,' said Arthur, 'this piece of chicken?'

Slartibartfast glanced at him gravely.

'Toy with it,' he said, 'toy with it.'

He demonstrated with his own piece.

Arthur did so, and felt the slight tingle of a mathematical function thrilling through the chicken leg as it moved four-dimensionally through what Slartibartfast had assured him was five-dimensional space. ...

'I don't like this wine very much,' said Arthur sniffing it.

'Well send it back. It's all part of the mathematics of it.'

Arthur did so. He didn't like the topography of the waiter's smile, but he'd never liked graphs anyway.

'Where are we going?' said Ford.

'Back to the Room of Informational Illusions,' said Slartibartfast, rising and patting his mouth with a mathematical representation of a paper napkin, 'for the second half.'

Excerpted from

Life, the universe and everything

by Douglas Adams

To Donna, who helped pay for this
in more ways than one,
and to my family and friends
for their support and humor

Abstract

Vibration analysis of rotating machinery is an important part of industrial predictive maintenance programs. With it, wear and defects in moving parts can be discovered and repaired before the machine breaks down, thus reducing operating and maintenance costs.

One method of vibration analysis is known as *order tracking*. This is a frequency analysis method that uses multiples of the running speed (orders), instead of absolute frequencies (Hertz), as the frequency base. Order tracking is useful for machine condition monitoring because it can easily identify speed-related vibration, such as shaft defects and bearing wear, even at high orders. To use order tracking analysis, the vibration signal must be sampled at constant increments of shaft angle. Conventional order tracking methods use specialized analog hardware to sample at a rate proportional to the shaft speed. A newer, computed order tracking method has been introduced which samples at a constant rate (i.e., uniform Δt) and then uses software to resample the data at constant angular increments.

This study examines which factors and assumptions inherent in the new order tracking method have the greatest effects on accuracy. Both classical and computed methods have been evaluated and compared using a digital simulation. It was found that the method is extremely sensitive to the timing accuracy of keyphasor pulses, and improvements in spectral accuracy would result if cubic spline interpolation were implemented instead of linear interpolation.

Table of Contents

1. Introduction 1

2. Background & Theory 3

2.1. Purposes of Vibration Analysis 3

2.2. Contemporary Analysis Methods 4

2.2.1. The Keyphasor 4

2.2.2. Time-Domain Analysis of Vibration Data 5

2.2.2.1. Vibration Severity 6

2.2.2.2. Lissajous Measures 6

2.2.2.3. Average Shaft Centerline Position 8

2.2.3. Frequency-Domain Analysis of Vibration Data 9

2.2.3.1. Common Bearing Defect Signature Patterns 10

2.2.3.2. Single Spectrum 10

2.2.3.3. Stacked Spectra, Cascade or Waterfall Plot 11

2.2.3.4. Bode Plots 13

2.2.3.5. Periodic averaging 14

2.2.3.6. Order Tracking 14

2.3. Theory 16

2.3.1. Fast Fourier Transform 16

2.3.2. Traditional Means of Order Tracking 17

2.3.3. Hewlett Packard Method of Computed Order Tracking 18

2.3.4. Bruel & Kjaer Method of Computed Order Tracking 23

3. Equipment & Procedures 24

3.1. Equipment 24

3.2. Use of Published Algorithms & Data 24

3.2.1. Hewlett Packard Method of Computed Order Tracking 24

3.2.2. Digital Filter Coefficients 25

3.2.3. Block Cubic Interpolation 25

3.2.4. Fast Fourier Transform 25

3.3. Procedure 25

4. Results & Discussion 28

4.1. Processing Methods 28

4.2. Similarity Factor 28
4.3. Comparison of Methods 29
4.4. Factors Affecting Accuracy of the HP Method 31
4.4.1. Effects of the Keyphasor 31
4.4.1.1. Keyphasor Timing Accuracy & Resolution 31
4.4.1.2. Using Multiple Keyphasor Pulses per Revolution 32
4.4.1.3. Keyphasor Noise 35
4.4.2. Effects of Filtering 36
4.4.2.1. Digital Filtering 36
4.4.2.1.1. Filter Design 37
4.4.2.2. Interpolated Digital Filtering 38
4.4.3. Effects of Rotation Speed & Acceleration 41
4.4.3.1. Assumption of Consistent Shaft Angle 41
4.4.3.2. Misrepresentation of Frequency Content 42
4.4.3.3. Attenuation of High Orders 43
4.4.3.4. Varying Acceleration 44
4.4.4. Effects of Interpolation Method 46
4.4.4.1. Linear Interpolation 48
4.4.4.2. Quadratic Interpolation 50
4.4.4.3. Piecewise Cubic Interpolation 52
4.4.4.4. Blockwise Cubic Spline Interpolation 54
4.4.4.5. Interpolation Summary 56
4.4.5. Considerations Regarding Noise 57
4.4.5.1. Sources of Noise 58
4.4.5.2. Effects of Noise on Accuracy of the HP Method 58
4.4.5.3. Effects of Noise on Choice of Interpolation Method 60
4.4.6. Effects of Data Block Size 60
5. Summary & Conclusions 62
6. Recommendations for Future Development 64
7. Bibliography 65
8. Appendices 67
8.1. Fundamental code 67
8.1.1. Main program 67

8.1.2. Hewlett Packard Method of Computed Order Tracking 71
8.1.3. Subroutine for Resampling Data 72
8.1.4. Subroutine for Determining Resample Times 73
8.2. Sample Initialization File 76

List of Tables

Table 1: Similarity Comparison 32

Table 2: Amplitude Comparison of Interpolation Methods 52

Table 3: Spectral Comparison of Interpolation Methods 52

List of Figures & Diagrams

Figure 1: Diagram of rotating machine 4

Figure 2: Illustration of keyphasor transducer mounting 5

Figure 3: Example of Lissajous plot 7

Figure 4: Illustration of XY vibration sensors 7

Figure 5: Example of average shaft centerline plot 8

Figure 6: Example of single spectrum presentation 10

Figure 7: Simulation of runup analyzed using traditional techniques 11

Figure 8: Example of Bentley Nevada format for Bode plot 12

Figure 9: Simulation of runup analyzed using order tracking techniques 14

Figure 10: Equipment used for traditional order tracking 16

Figure 11: Equipment used for HP's computed order tracking 17

Figure 12: Flowchart of computed order tracking algorithm 19

Figure 13: Example of resampled signal 20

Figure 14: Flowchart of simulation program 25

Figure 15: Flowchart of simulation procedure 26

Figure 16: Comparison of classical order tracking to other methods 29

Figure 17: Effects of keyphasor resolution 30

Figure 18: Effect of using multiple keyphasors with linear acceleration 32

Figure 19: Effect of using multiple keyphasors with nonlinear acceleration 33

Figure 20: Effect of adding random error to keyphasor signal 34

Figure 21: Digital filter frequency response for 700 rpm trigger speed 36

Figure 22: Family of digital filter coefficient curves 37

Figure 23: Removal of high orders using digital filtering 38

Figure 24: Detailed examination of filter effects 38

Figure 25: Increased rotation speed decreases keyphasor accuracy 39

Figure 26: Spectrum at 400 rpm during high acceleration 40

Figure 27: Spectrum at 1000 rpm during nonlinear acceleration 42

Figure 28: Spectrum at 400 rpm during nonlinear acceleration 43

Figure 29: Detail of linear interpolation near 1000 rpm 46

Figure 30: Spectrum from linear interpolation at low and high speeds 46

Figure 31: Detail of quadratic interpolation near 1000 rpm 47

Figure 32: Spectrum from quadratic interpolation at low and high speeds 48

Figure 33: Detail of piecewise cubic interpolation near 1000 rpm 49

Figure 34: Piecewise cubic interpolation spectrum at low and high speeds 50

Figure 35: Detail of blockwise cubic interpolation neat 1000 rpm 51

Figure 36: Spectrum from block cubic interpolation at low and high speeds ... 51

Figure 37: Effects of quantization and digital word length 54

Figure 38: Spectrum showing effects of random noise 55

Symbols, Nomenclature & Abbreviations

a_n	curve fit coefficients
b_n	curve fit coefficients
$\Delta\Phi$	angular separation of keyphasor pulses
$\Delta\theta$	angular separation of order data samples
Δt	elapsed time between one sample and the next
Φ	shaft angle
f	frequency
f_c	cutoff frequency
f_c'	cutoff frequency (in order domain)
f_s	sampling frequency
$H(f)$	function of frequency
$h(t)$	function of time
k	an integer
N	number of samples in data block
π	ratio of a circle's circumference to its diameter, approximately 3.14159
θ	shaft angle
S	similarity factor
t	time
t_n	arrival time of n^{th} keyphasor
x	independent variable
y	dependent variable
[n]	bibliographical reference
1x, 2x,...	one times the shaft speed, two times the shaft speed, etc.
ADC	analog to digital converter
B&K	Bruel & Kjaer
COT	computed order tracking
DFT	discrete Fourier transform
Eqn.	Equation
FFT	fast Fourier transform
Fig.	Figure
HP	Hewlett Packard
Hz	Hertz, cycles per second

1 Introduction

In industry, the cost of operating rotating machinery increases dramatically when premature breakdowns occur. To lower costs, it is important to monitor machinery condition to predict when service is due. One common approach is to use vibration analysis because it can provide extensive insight, is nonintrusive, and can be done while the machinery is operating [9]. This vibration analysis can be done in either the time or frequency domain.

Time domain analysis can indicate the motion of particular machine elements beneath the transducer, or overall vibration levels. Frequency domain analysis can reveal what frequencies are contained in the vibration signal. From this, it is possible to determine what machine components are responsible for the vibration. This information can be used to help plan repairs.

A special case of frequency analysis is called *order tracking*, and uses multiples of the running speed (called orders) instead of absolute frequency units (like Hertz) as a frequency base. This can be helpful because many vibrations in rotating machinery are usually directly related to the rotation of the main shaft.

Historically, order tracking has been performed using analog equipment to directly sample the vibration signals of rotating machinery at constant increments of the shaft angle [13]. The associated cost and complexity remain the major stumbling blocks that prevent the widespread use of this technique. This analog approach is also error prone; the equipment used for analog order tracking is known to have problems following changing shaft speeds [16].

In 1989, Hewlett Packard (HP) introduced a computational method for performing order tracking analysis that reduced the dependency on specialized equipment and claimed to give accuracy superior to analog methods [14]. Soon after, Bruel & Kjaer (B&K) introduced a different computational method of achieving essentially the same result, making the same claims as HP regarding superiority over analog approaches [20]. HP uses a method of resampling by interpolation filtering, while B&K uses a method of resampling by decimation and interpolation.

Some information has been published regarding how computed order tracking (COT) works [7, 12, 13 & 14] and why it is better than analog methods, but no literature exists to explain how the assumptions and approximations inherent in these methods affect accuracy. This study examines the HP method of COT in detail to determine which factors affect its accuracy. Details regarding the precise nature of the B&K method appeared too late to be

included. Future development may see this method added to the simulation and compared to others quantitatively.

With the anticipation of a real-time implementation of this method, it is necessary to find which factors greatly affect accuracy to ensure that computational power is not wasted. The computational process must keep spectral noise to a minimum so that faint vibration signals from small machine components can be distinguished in the presence of stronger ones. By determining the most important accuracy factors, minimum hardware requirements may be determined. In addition to the hardware requirements, this study indicates which software approaches are most likely to yield the best results.

To learn how the HP method works, a full numerical simulation was created and implemented using a digital computer. By using a simulation, all factors can be held constant except those of interest, allowing determination of the factors that have the greatest effects on accuracy. Incorporating a simulated signal ensures that the raw signal data is identical for every test.

2 Background & Theory

2.1 Purposes of Vibration Analysis

The purpose of industrial maintenance is to keep production machinery operating effectively and reliably at the lowest possible cost. When a machine unexpectedly breaks down, the owner loses money in two ways: production (i.e. revenue) is lost, and money is spent on the necessary repairs. Such emergency repairs are usually more expensive than if the same repair had been planned.

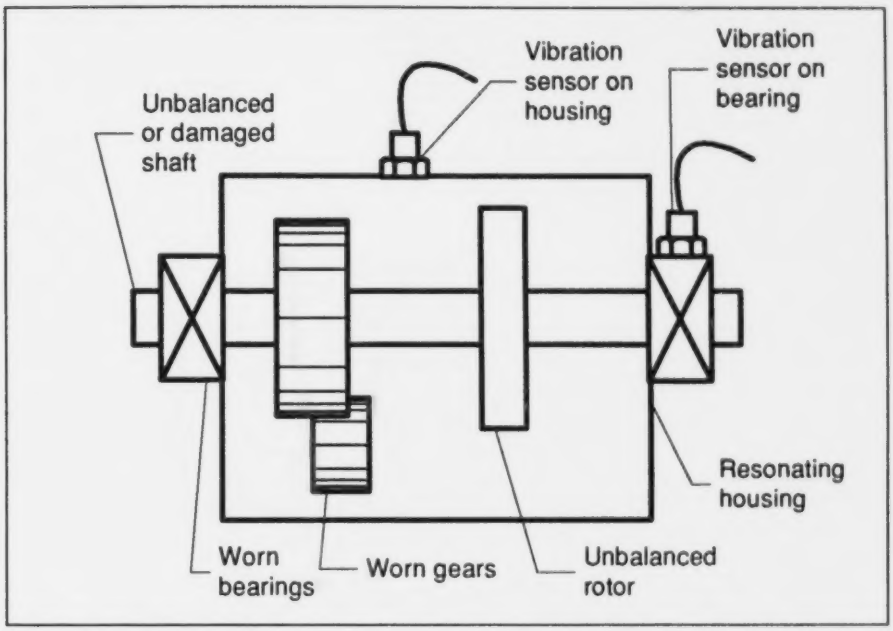
For these reasons, most industries plan their maintenance. *Preventive maintenance* is a simple approach that schedules repairs based on the history of equipment failure and machine manufacturers' recommendations. This system is straightforward but not optimal. Often, money is wasted when parts are replaced unnecessarily, and there is no guarantee that a part will last until its replacement date. When a part fails before its scheduled replacement, the event being avoided (an unexpected breakdown) occurs.

A better method involves monitoring machine condition and scheduling repairs just before failure. This would allow the owner to get full money's worth from parts purchased, and to avoid machine failure. Accurate determination of machine condition can also reduce downtime by allowing maintenance to be scheduled and by providing longer intervals between servicing. This is an approach known as *predictive maintenance.*

Vibration analysis of rotating machinery can be an important part of industrial predictive maintenance programs, since it allows the condition of operating machinery to be monitored without shutdown or disassembly of the machine. It is possible to assess machine condition by monitoring vibration levels because the moving parts in any machine cause vibrations and as these parts wear, their vibration levels increase. Figure 1 shows typical running problems with common machine elements and usual transducer placement. The vibration data is used to decide what, if anything, is wrong with the machine. If excessive wear is detected, the machine can be removed from service before it fails. Such analysis can ensure safer operation of the machinery by not allowing wear-induced catastrophic failure to occur, and make operation less costly by allowing better management of the repair project.

Simple approaches to vibration analysis measure total vibration levels; when the level becomes excessive, the machine is removed from service. Often, however, critical small components such as bearings or gear teeth fail before

total vibration levels become excessive, and extensive damage occurs before the problem is recognized.



► Figure 1: Diagram of rotating machine

In normal practice, some combination of the two methods (preventive and predictive) is used. Those machines that are critical to production, pose safety risks, or are very expensive to repair are maintained using predictive techniques such as vibration analysis. The inexpensive and noncritical machinery, often called *balance-of-plant machinery*, are maintained using preventive methods.

2.2 Contemporary Analysis Methods

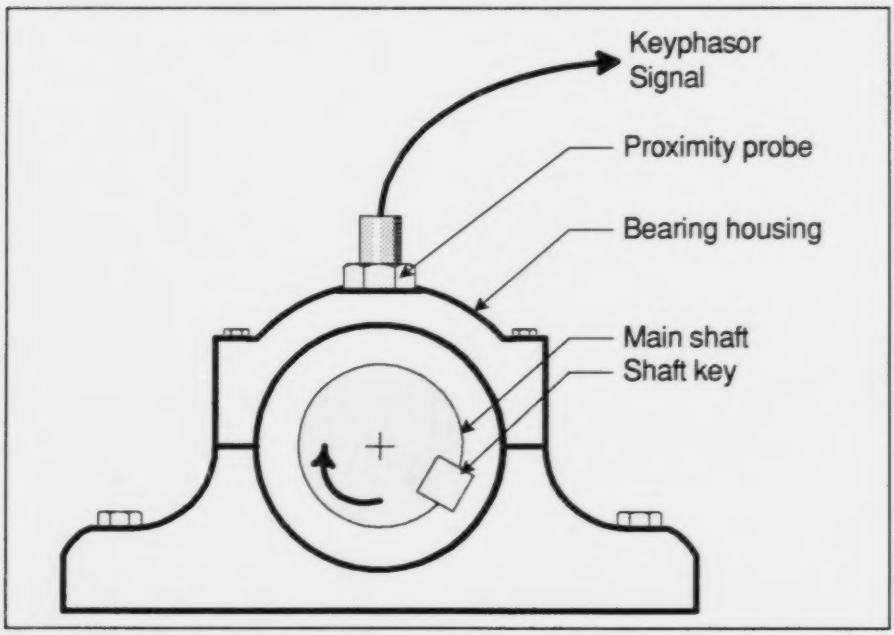
While the simplest and least expensive form of vibration monitoring uses subjective tests (the human senses), more specific and repeatable information can be obtained using objective tests (measurements). The most common measurements are made in either the time or frequency domain.

2.2.1 The Keyphasor

A common component of most analysis methods is that of a keyphasor. It is defined as a once-per-shaft-revolution event used to measure shaft speed and as a reference for measuring vibration phase angle. The term *keyphasor* was

coined by Bentley Nevada [2]. The keyphasor transducer is typically a proximity probe, optical pickup or magnetic pickup.

Vibrations in a machine can have a certain phase relationship, i.e. a difference in start time of the vibration cycle of each component. To give a common reference point, one position on a rotating shaft of the machine is chosen. All other vibration components are timed according to the repeated passage of this point. The most common reference is a shaft key because it is usually accessible and gives a strong signal, as shown in Figure 2. Thus, because the measurement is taken from a shaft key and because it is the primary phase marker, the signal is called the keyphasor.



► Figure 2: Illustration of keyphasor transducer mounting

2.2.2 Time-Domain Analysis of Vibration Data

Time-domain analysis methods directly examine the transducer signal, although some filtering may be applied. Either one or two vibration signals and a tachometer (keyphasor) signal may be used and analyzed using either analog or digital instrumentation. Some common methods of presenting the data are discussed below.

2.2.2.1 Vibration Severity

Vibration severity measurements are a method of comparing the magnitude of a given vibration to a predetermined allowable limit [9]. One approach is to rectify the output of a vibration transducer and obtain a Root-Mean-Square (RMS) value of the signal. This provides a coarse measurement of how bad the vibrations (and therefore wear) are in a given machine. The machine is removed from service and repaired when the vibrations reach the preset limit. Since the vibration severity measurement is a single number, the data are usually presented in some sort of trend graph, usually showing an increase in vibration severity over time.

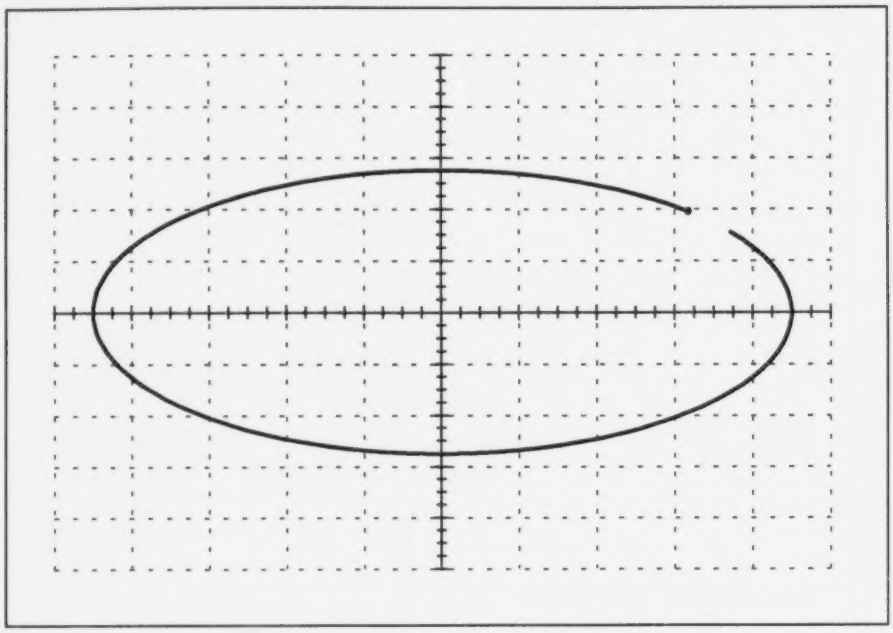
Vibration severity measurements have two drawbacks. First, they do not suggest the source of the vibrations, so specific repairs are hard to plan. Second, critical small parts such as gear teeth or bearings may fail without severely raising vibration levels because their contribution to the total vibration level is small. After these components fail, the resulting damage to other moving parts will raise vibration levels (along with repair cost) until the vibration limit is reached. A better method would detect these failures before more serious damage occurs.

2.2.2.2 Lissajous Measures

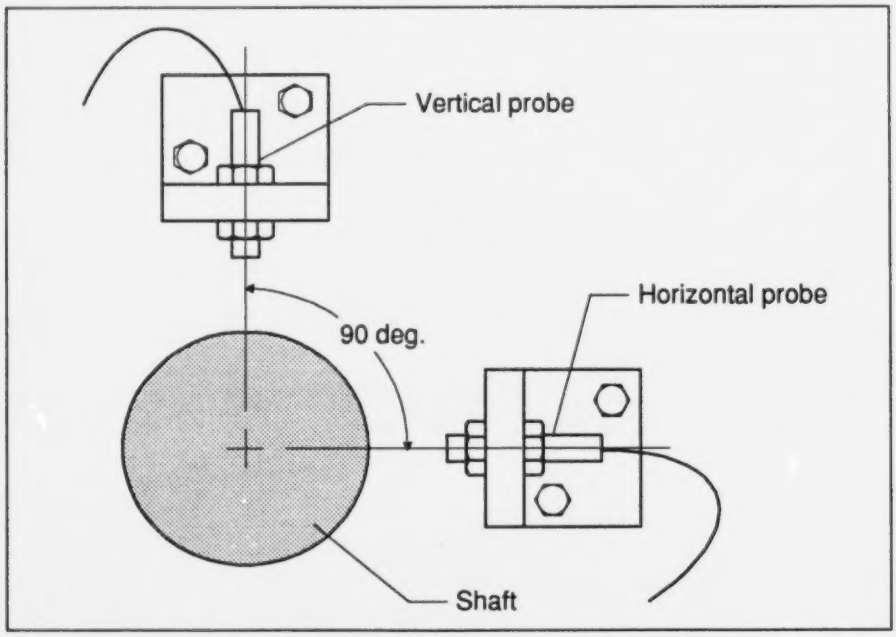
More commonly known by the Bentley Nevada term *orbit plots* [2], these XY graphs show the path of the shaft center during one rotation, as in Figure 3. An orbit can be observed on an oscilloscope (an analog device) or a digital analyzer. Other names for this presentation of data are precessional motion, orbital motion, or Lissajous presentation [6, 8].

Whether the display is analog or digital, Lissajous measures use two transducers at right angles to each other and a keyphasor sensor, as shown in Figure 4. Usually used for such operations as shaft balancing, Lissajous measures can also give a reasonable indication of bearing wear.

An additional use of this type of measurement, if several bearings are monitored along the length of the shaft and timed by one keyphasor, is to observe the mode shape of the shaft vibration.



► Figure 3: Example of Lissajous Plot



► Figure 4: Illustration of XY vibration sensors

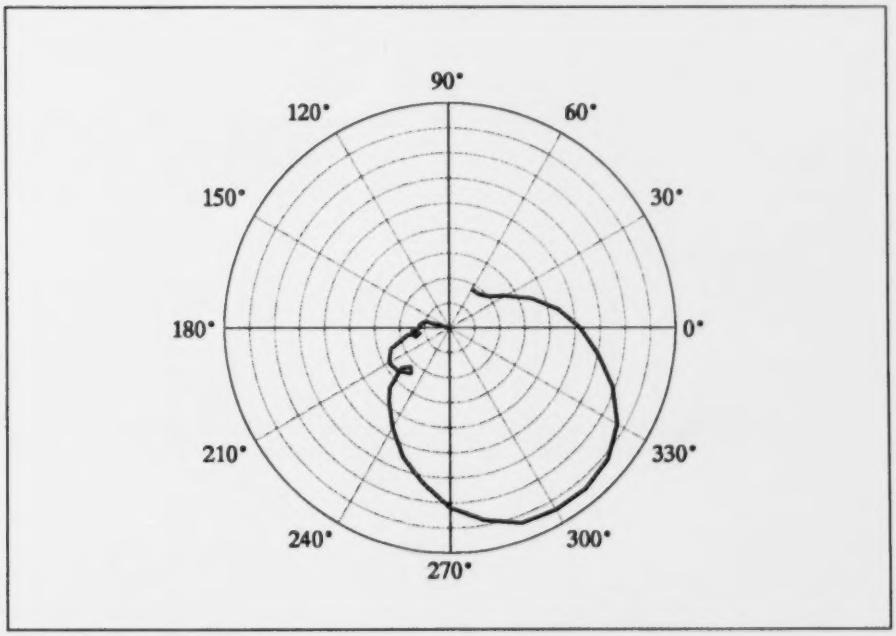
2.2.2.3 Average Shaft Centerline Position

Presented using polar axes, these plots show the average shaft center position over the course of a run-up or rundown, as illustrated in Figure 5. Although they use the same sensor array as for Lissajous plots and show the same kind of data, they do so over a longer term. These plots can show the speeds at which vibration amplitudes are most severe, and to some extent, when the mode shapes of the vibrations change.

Bentley Nevada refers to these presentations by the general term *polar plots*, and defines them as,

> Polar coordinate representation of the locus of the 1X (or 2X,...) shaft vibration *vector* at a specific *lateral* shaft location with shaft rotative speed as a parameter. The polar plot is generated by *in-phase* and *quadrature* signals, usually during machine startup or coastdown. [2]

In this situation, the 1x (or 2x,...) vibration components from the raw data signal are found using specialized digital filters.



► Figure 5: Example of average shaft centerline position graph

2.2.3 Frequency-Domain Analysis of Vibration Data

The basis of frequency-domain analysis is the Fast Fourier Transform or FFT. The FFT is a numerical technique used to calculate the frequency content of discrete time-domain data. Analog time-domain data is converted to a digital format by sampling the data at constant time intervals, Δt. This analog data is normally passed through an analog filter prior to sampling to prevent aliasing, as will be explained in Section 2.4.2, Fast Fourier Transform.

Frequency-domain analysis shows frequency components of the vibration signal, helping pinpoint the causes of vibrations. Signature patterns in the spectra may identify common machine faults, discussed further in Section 2.3.2.1.1, Common Bearing Defect Signature Patterns. Some common methods of presenting this spectral information are outlined in the sections that follow.

2.2.3.1 Common Bearing Defect Signature Patterns

It is important to recognize that bearing defect frequencies of any type indicate a problem by their presence as much as by their magnitude. These frequencies simply should not exist in the spectra; if they do, there is a problem.

Defective rolling-element bearings generate two main types of frequencies: *natural* frequencies of bearing components and *rotational defect* frequencies. Natural frequencies of bearing components are excited when the rolling elements strike defects in the inner or outer races. These frequencies are a function of the bearing geometry, and are independent of running speed.

Equations have been developed [3] that relate specific bearing defects to specific rotational defect frequencies. The interesting feature of this type of vibration for order tracking analysis is that they are all a function of rotational speed. These defects can occur on any part of the bearing assembly (inner race, outer race, rolling elements or cage) and there are equations for each. Some publications exist [3] which have tabulated these defect frequencies for various makes and models of bearings.

Some of the main features of the signature frequencies are these:

a. bearing defect frequencies are one of only a few order-based phenomena which occur at non-integer (asynchronous) orders

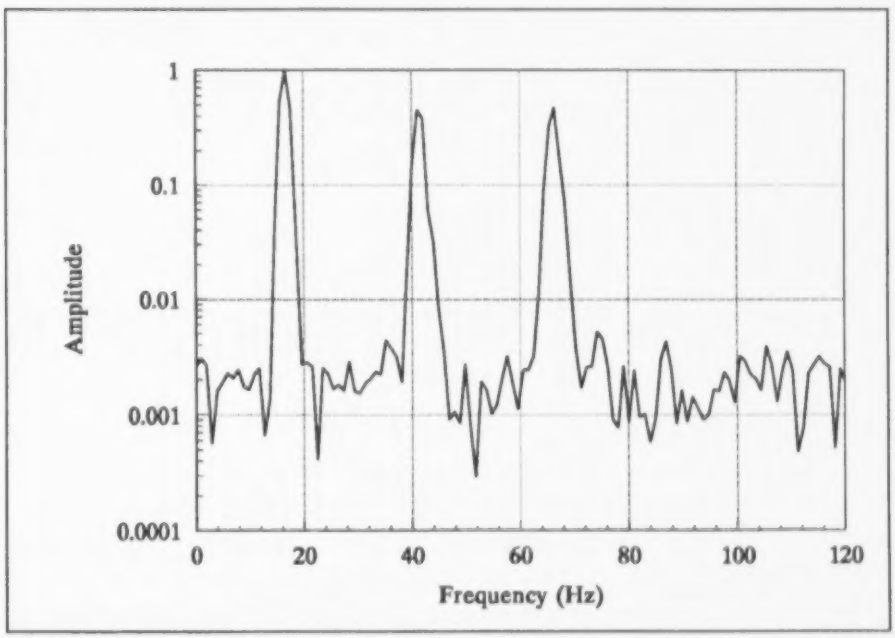
b. the amplitude of the outer race defect frequency is higher than the inner race frequency because the transducer is closer to the outer race

c. defects appear first on the races (generating the main spectral peaks) then on the rolling elements and the cage, causing smaller sidebands to appear around the main peaks.

In the numerous examples used by Berry [3], the defect frequencies are usually less than 10 orders, so the somewhat arbitrary choice of a 0-6 order range for this study is probably valid.

2.2.3.2 Single Spectrum

A spectrum usually shows frequency on the independent axis and amplitude on the dependent axis, as shown in Figure 6. Since the FFT output is complex, the amplitude is found by calculating the vector length of the real and imaginary components. A single spectrum is used for machines running at a fixed speed, or for close examination of the frequency content of a signal.



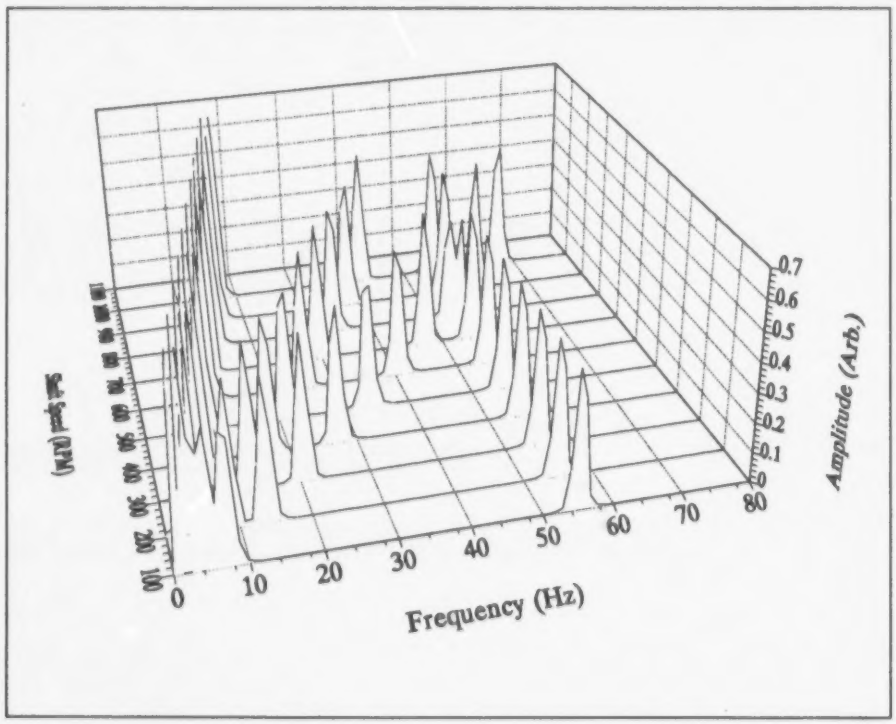
► Figure 6: Example of single spectrum presentation

2.2.3.3 Stacked Spectra, Cascade or Waterfall Plot

It is common to analyze rotating machinery by performing vibration analysis while the machine is speeding up or slowing down. When the machine starts from rest and accelerates smoothly while vibration signals are monitored, the process is called a *runup*. When the machine starts from a higher speed and

decelerates, the process is called a *coastdown* or *rundown*. Performing vibration analysis during a runup or rundown can reveal problems that might not appear if the machine operated at steady state. Runups and rundowns are especially useful when combined with order tracking analysis, as order-based phenomena become readily apparent (as in Figure 9). For machine runups and rundowns, it is useful to compare the spectra obtained at various machine running speeds. To this end, individual spectra are often stacked in one plot, allowing any patterns within the spectra to become more easily discernible.

Figure 7 shows a simulation of a machine run-up analyzed using traditional frequency-domain techniques. As the machine passes a certain threshold of speed (a trigger speed), a block of time-domain vibration samples is taken and an FFT of those samples is calculated. In this plot, the triggers started at 100 rpm and occurred every hundred rpm up to and including 1000 rpm. The machine had three shaft-related vibrations at 1x, 2.5x and 4x the shaft speed, and a fixed-frequency component at 55 Hz. In Figure 7, notice the corresponding three oblique patterns of peaks and the one vertical pattern.



► Figure 7: Simulation of runup analyzed using traditional techniques

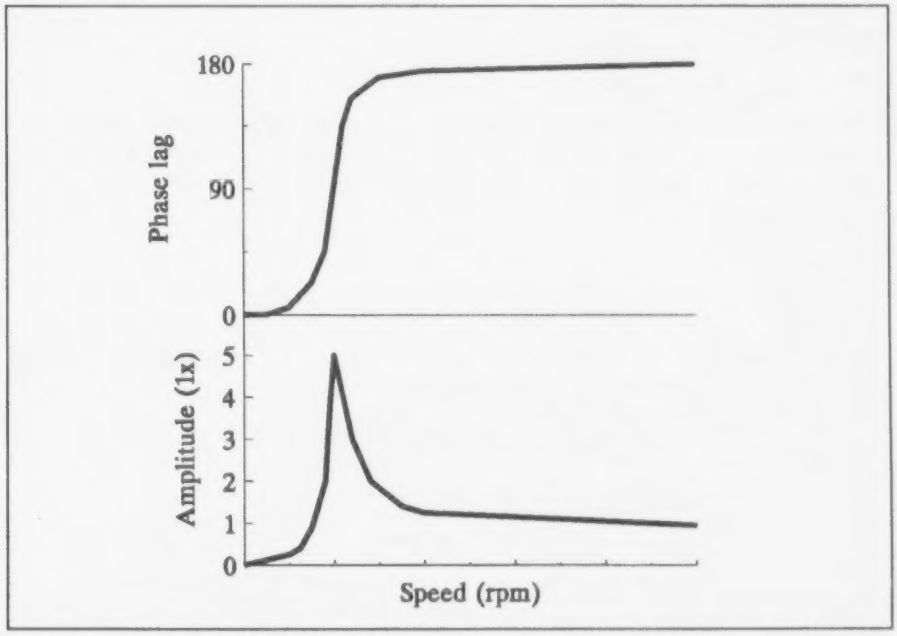
This type of data presentation displays information about shaft-related frequencies, but not well. The peak pattern for any high orders would be hard to distinguish as the slope would be too flat, and there would not be enough peaks. If the sample rate was changed to show high orders well, distinction (resolution) between the low orders would be lost.

Bentley Nevada [2] distinguishes a cascade plot from a waterfall plot; the cascade plot compares the information using rpm on one axis, while the waterfall plot uses time. Occasionally, one axis will be skewed to give the plot an isometric appearance; this modified plot is called a *raster plot*.

2.2.3.4 Bode Plots

The classical Bode plot is a special case of the single spectrum presentation that shows two graphs at once. One shows amplitude vs. frequency, and the other shows phase vs. frequency, yielding insight into phase relationships between frequency components.

Bentley Nevada's interpretation of the Bode plot is quite different (see Figure 8) ; according to them, it is a



► Figure 8: Example of Bentley Nevada format for Bode plot

> graph in Cartesian format representing the 1X vibration vector response as a function of shaft rotative speed. One Y axis represents 1X amplitude, a second Y axis represents phase lag angle, and a common X axis represents shaft rotative speed. Sometimes called an imbalance response plot. Also used for 2X, 3X,... vibration response vectors. [2]

Bentley Nevada uses their version of Bode Plots for balancing rotating machinery. A Bode plot gives a complete snapshot of the rotor response. The phase angle plot shows the position and relationship of the rotor's *heavy spot* (where too much weight exists) and its *high spot* (where weight must be added). The amplitude plot shows the resonance frequency(ies) and amplitude(s).

2.2.3.5 Periodic averaging

Before the introduction of more modern computational methods, a simplified form of extracting order data was developed called *periodic averaging* [11], which was normally done as a postprocessing technique. In this method, a cycle period is identified (usually one rotation of the shaft) and then the recorded data signal is resampled a predetermined number of times per revolution. When data from several periods are obtained they are averaged to produce one average period. An FFT of this data produces an order spectrum.

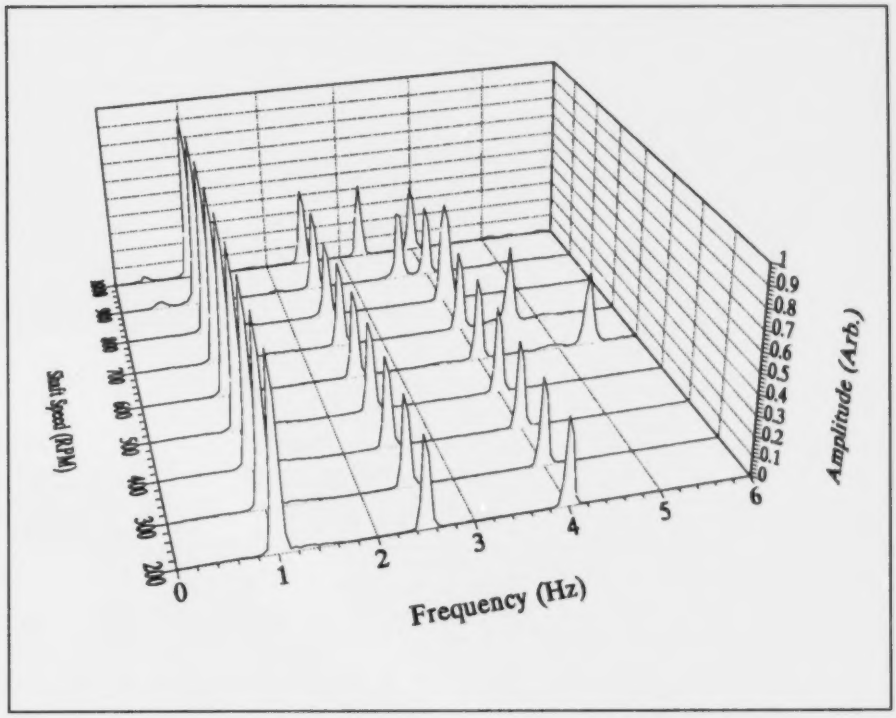
Aside from concerns regarding frequency resolution, the primary disadvantage of this method is the loss of all non-order data. Any time-dependent data is assumed to be random, and is removed by the averaging process; other than such "random noise", all that remains in the spectrum is order-related data.

2.2.3.6 Order Tracking

Vibrations directly related to shaft speed are said to occur at certain *orders*. Vibrations from the blades of a 12-blade fan, for example, would be expected to occur at a frequency of 12 orders. Other machine elements have their own signature frequencies, usually related directly to the shaft speed. In discussing analysis of gearbox vibrations, Tan and Mathew [16] note that when speed varies, a time base is inadequate for describing the periodicity of vibration, but they recognize that it is possible to describe the vibration as "a periodic function of the shaft angular position". This is the basis for order tracking analysis.

Figure 9 shows the same run-up as shown in the frequency domain earlier, but performed by order tracking analysis. Order related vibrations are easily

distinguished as vertical patterns of peaks; note that all orders have equal frequency resolution. Notice also that the fixed-frequency component ceases to be a straight pattern of peaks, and so the fixed-frequency vibration information is not obvious. Thus, order tracking does not replace other methods, but adds another way to look at vibration data.



► Figure 9: Simulation of runup analyzed using order tracking techniques

Order tracking is a frequency-domain approach, but it relates vibration amplitudes to orders instead of absolute frequencies. In this way, bearing faults, gear tooth wear and other vibrations associated with the running speed are easily identified.

This change of units is accomplished by understanding a property of the Fourier transform equations, [18]

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

If the quantity t in the above equations is measured in seconds, then the quantity f will have the units of cycles per second (Hz), but these equations are not restricted to these units. If $h(t)$ is a function of revolutions, then $H(f)$ will have the units of cycles per revolutions. The units of frequency measured in cycles per revolution are called orders. To summarize: in traditional frequency analysis, data are transformed from the time domain into the frequency domain; in order tracking analysis, data are transformed from the angle domain into the order domain.

2.3 Theory

2.3.1 Fast Fourier Transform

The basis of frequency-domain analysis is the Discrete Fourier Transform or DFT. The DFT is a numerical technique used to calculate the frequency content of discretely sampled time-domain data. A more computationally efficient form of the DFT is the Fast Fourier Transform, or FFT.

Real vibration signals usually contain a very wide bandwidth; that is, frequencies in the signal can range from zero to very high frequencies. In vibration analysis, the user specifies the highest frequency of interest, and employs an analog lowpass filter to remove all higher ones. This highest frequency is known as the cutoff frequency, f_c. When sampling a signal at constant intervals, there is a limit on the maximum length of the sample interval, determined by the *Nyquist critical frequency*. The Nyquist critical frequency is twice f_c, the highest frequency in the signal. This limit ensures that at least two samples of each cycle of the highest frequency are taken. The sampling rate, f_s, is set at more than twice the highest frequency, ensuring that it will be sampled at least twice per cycle, i.e.

$$f_s \geq 2f_c$$

If the signal is sampled slower than this limit, a phenomenon known as *aliasing* occurs. Aliasing results in high frequencies "folding back" and appearing as lower frequencies in the signal. Once a signal has been aliased by discrete sampling, there is little that can be done to detect the aliasing or remove it. The solution is the prevention of aliasing, accomplished by analog lowpass filtering of the signal as mentioned above and sampling faster than the Nyquist limit.

The FFT generates a discrete frequency spectrum. If a periodic signal like a sine wave is sampled so that an integer number of periods fill the data block, then the FFT will produce a spectrum with amplitudes of zero for all frequencies except the frequency of the sine wave. However, obtaining an integer number of periods of a vibration signal rarely occurs. If a non-integer

number of periods is sampled, energy is spread across the spectrum around the main peak. This is a phenomenon known as *leakage*, and can be seen in Figure 6, where the central peaks are broadened instead of being thin spectral lines.

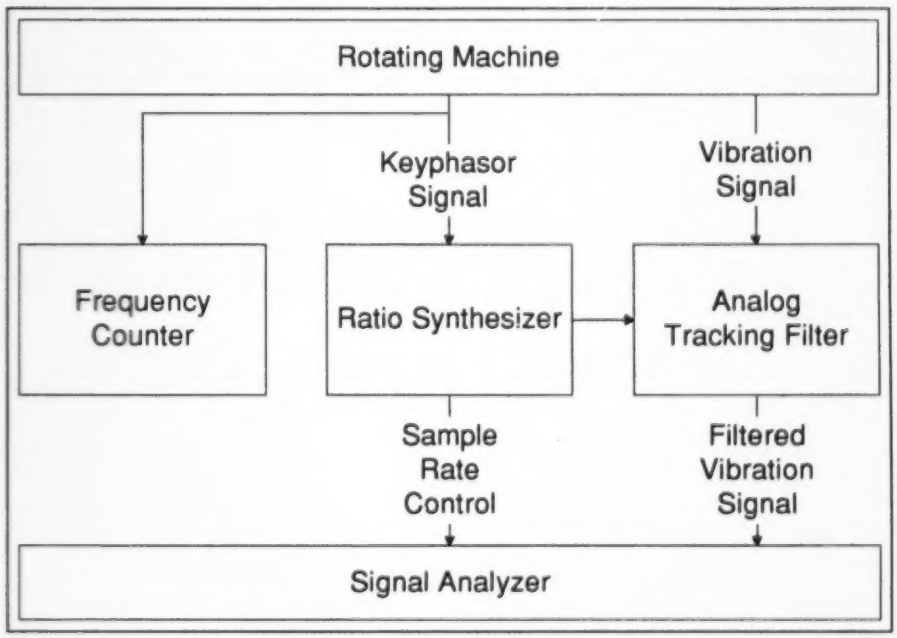
Leakage can be reduced by applying a *data window* to the data block prior to calculation of the FFT. Most data windows modify the data block so that it starts and ends with small amplitude signals. The data window known as a *Hanning* window [21] was used throughout this study. This window has the form

$$x(t) = \frac{1}{2} - \frac{1}{2}\cos\frac{2\pi t}{T_c} \quad 0 \le t \le T_c$$

where T_c is the total length of the data block and t is any given time within the data block.

2.3.2 Traditional Means of Order Tracking

Traditional order tracking attempts to directly sample the analog signal at constant $\Delta\theta$ using extra analog instrumentation, namely a ratio synthesizer and an anti-aliasing tracking filter, as shown in Figure 10. A frequency counter may also be included to monitor the shaft speed.



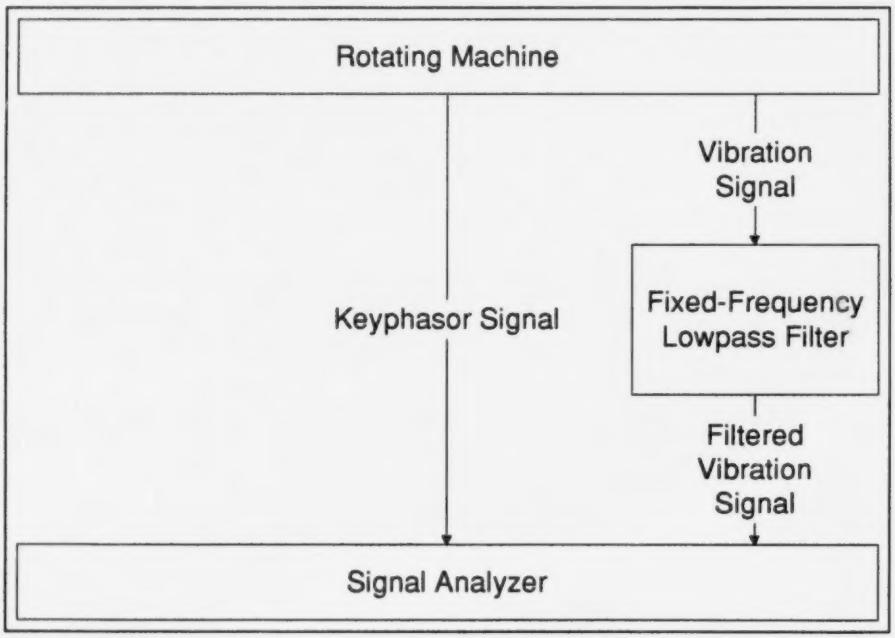
► Figure 10: Equipment used for traditional order tracking

The ratio synthesizer generates a signal proportional to the shaft speed of the machine. This output is used to control the sampling rate of the vibration signal and the cutoff frequency of the analog tracking filter, which is a lowpass filter with an adjustable pass band. An FFT is calculated using the data sampled at constant $\Delta\theta$ (angle domain samples) and the resulting spectrum is an order spectrum.

2.3.3 Hewlett Packard Method of Computed Order Tracking

Before the computational methods were introduced, order tracking analysis was performed using specialized analog hardware (illustrated in Figure 10) to change the sampling rate in real time. This method attempted to directly take samples at constant $\Delta\theta$ increments.

Hewlett Packard introduced a digital method which can computationally *resample* a constant Δt signal to provide the desired constant $\Delta\theta$ data, based on a keyphasor signal. This method is called Computed Order Tracking (COT). In contrast to the traditional method, COT is almost fully digital. Shown in Figure 11, The vibration signal passes through a fixed frequency lowpass filter, and is sampled at constant increments of time, Δt.



► Figure 11: Equipment used for HP's computed order tracking

To this point, the method resembles traditional frequency analysis more than order tracking. However, once the signal has been sampled, it is resampled by software using the tachometer (or keyphasor) signal to extract signal amplitudes at constant $\Delta\theta$. In Figure 11, the item labeled "signal analyzer" represents whatever device is used for data acquisition and signal processing. This could be a specialized piece of equipment or a high speed digital computer with data acquisition hardware.

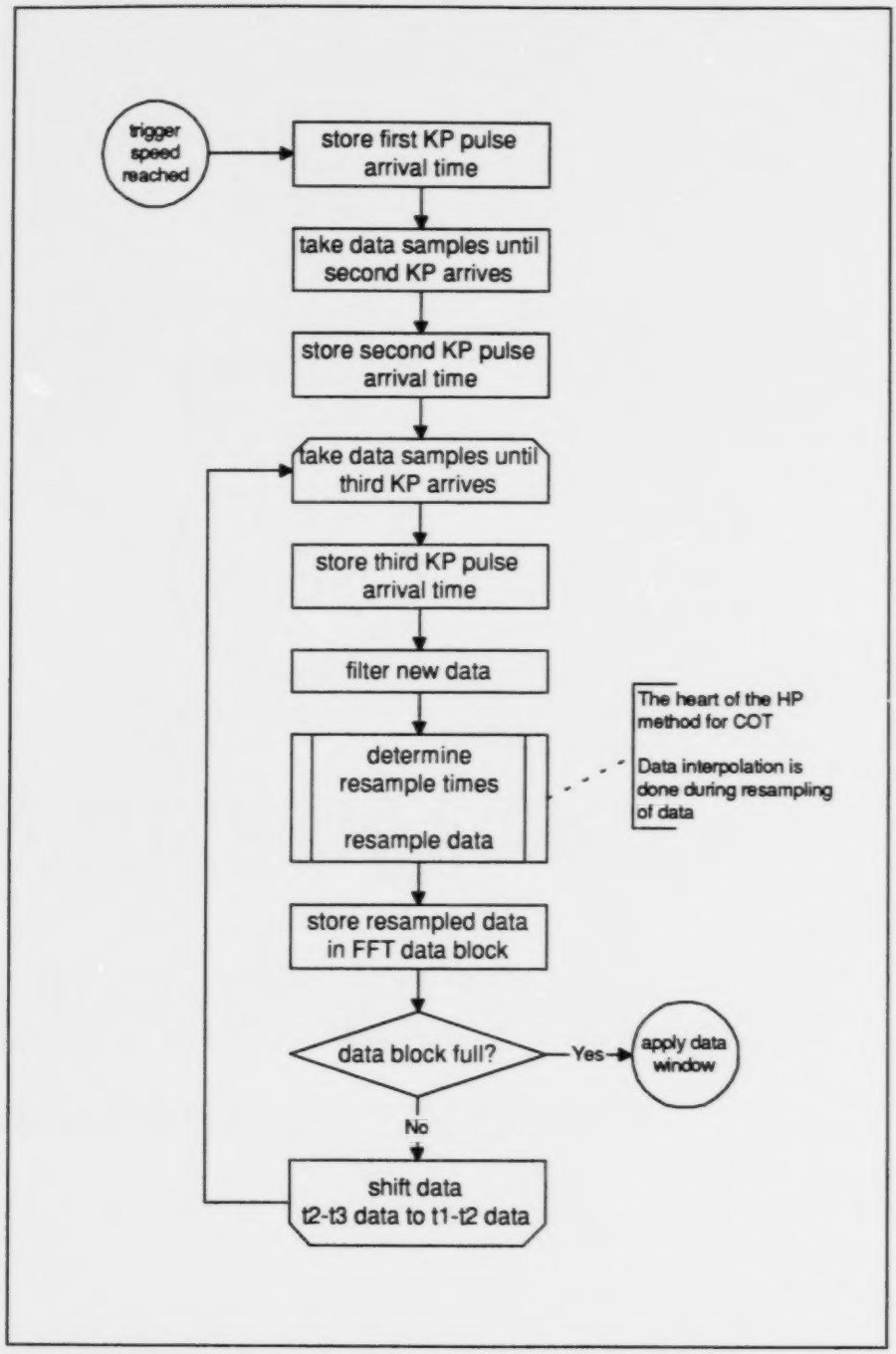
The analog lowpass filter is often built in to the signal analyzer or data acquisition system, and is inexpensive. In addition, it is likely to do a better job of filtering at its cutoff frequency than a tracking filter set at the same cutoff frequency. This is because the design of a variable (tracking) filter requires that compromises be made which reduce the effectiveness of the filter at a specific frequency in order to improve its performance at some other frequency. The goal is to achieve consistent filter performance across the range of adjustment. In contrast, a single-frequency filter can be optimized to perform extremely well at one particular frequency.

The process of computed order tracking is illustrated in Figure 12, although some detail has been omitted for clarity.

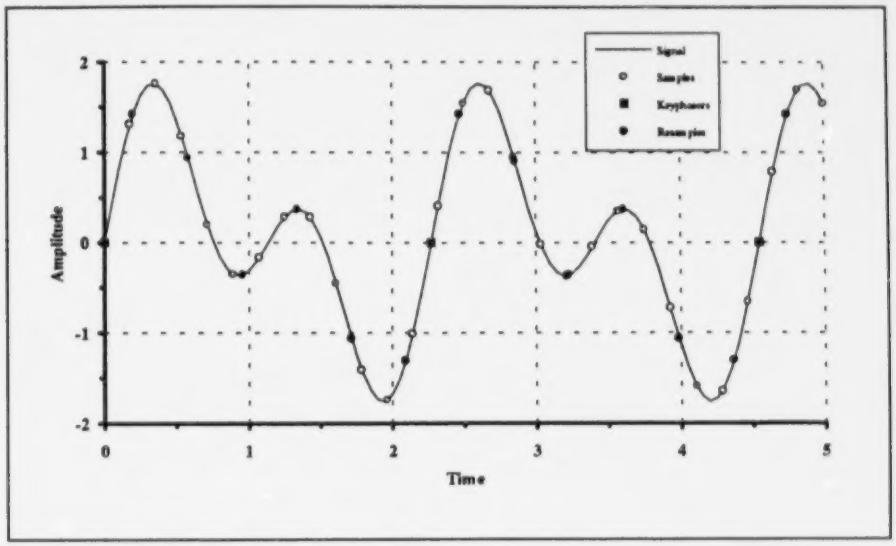
From the signal point of view, Figure 13 shows an arbitrary simple vibration signal containing only order data and delimited by three keyphasor pulses. Once the signal has been digitized by sampling at constant Δt, these data are resampled at constant increments of shaft angle. The timing of the resample points is based on the duration between keyphasor pulses.

For example, assume that it is desired to resample these data at six points per revolution. In Figure 13, the resamples are shown without error. Notice that the resamples fall on the same place on each wave (i.e. peak, trough) independent of where the actual time-based samples were taken. It will be shown that the better these values can be interpolated, the better the results will be.

When resampling occurs, two distinctly separate estimation processes occur. The first is the correct placement of the resamples on the independent (time) axis. This is the process of *determining the resample times*. The second is the correct placement of the resamples on the dependent (amplitude) axis. This is the *interpolation process*. Determining the precise resample times is critical for the interpolation process — without precision here, the interpolation process has no hope of consistent accuracy.


trigger speed reached
store first KP pulse arrival time
take data samples until second KP arrives
store second KP pulse arrival time
take data samples until third KP arrives
store third KP pulse arrival time
filter new data
determine resample times
resample data
The heart of the HP method for COT
Data interpolation is done during resampling of data
store resampled data in FFT data block
data block full?
Yes
apply data window
No
shift data t2-t3 data to t1-t2 data

► Figure 12: Flowchart of computed order tracking algorithm



► Figure 13: Example of resampled signal

For computed order tracking, the resample times are computed assuming a constant angular acceleration with time, t, so the shaft angle, Φ, can be described by the quadratic equation:

Eqn. 1:
$$\Phi(t) = b_0 + b_1 t + b_2 t^2$$

which, if solved for t becomes,

Eqn. 2:
$$t = \frac{1}{2b_2}\left[\sqrt{4b_2(\Phi - b_0) + b_1^2} - b_1\right]$$

The values of the coefficients b_0, b_1 and b_2 are found by fitting the keyphasor arrival times, which occur at known shaft angle increments[1],

Eqn. 3:
$$\Phi(t_1) = 0$$
$$\Phi(t_2) = \Delta\Phi$$
$$\Phi(t_3) = 2\Delta\Phi$$

[1] if the keyphasor is a once-per-shaft-revolution event, then the angular separation between keyphasors, $\Delta\Phi$, is known to be 2π radians.

to Eqn. 1, yielding

Eqn. 4:
$$\begin{Bmatrix} 0 \\ \Delta\Phi \\ 2\Delta\Phi \end{Bmatrix} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix} \begin{Bmatrix} b_0 \\ b_1 \\ b_2 \end{Bmatrix}$$

which is then solved for $\{b\}$,

Eqn. 5:
$$b_0 = \frac{t_1 \Delta\Phi}{\det}[t_3(t_1 - t_3) + 2t_2(t_2 - t_1)]$$

$$b_1 = \frac{\Delta\Phi}{\det}[t_1^2 - 2t_2^2 + t_3^2]$$

$$b_2 = \frac{\Delta\Phi}{\det}[-t_1 + 2t_2 - t_3]$$

$$\det = (t_2 - t_1)(t_3 - t_1)(t_3 - t_2)$$

Since the data are resampled after the arrival of each new keyphasor pulse, but the last three keyphasor pulses are used, the resample times are calculated only over the center half of the interval ($t_1..t_3$) to avoid overlap. Thus a limit is imposed on Φ,

Eqn. 6:
$$\frac{\Delta\Phi}{2} \le \Phi < \frac{3\Delta\Phi}{2}$$

For discrete resampling, it can further be specified that

Eqn. 7:
$$\Phi = k\Delta\theta$$

where $\Delta\theta$ is the desired angular spacing between resamples. Substituting this equation into Eqn. 6 yields values for k (a positive integer),

Eqn. 8:
$$\frac{\Delta\Phi}{2\Delta\theta} \le k < \frac{3\Delta\Phi}{\Delta\theta}$$

So finally, the equation for finding the resample times (Eqn. 2) becomes

Eqn. 9:
$$t_{k-k_{min}} = \frac{1}{2b_2}\left[\sqrt{4b_2(k\Delta\theta - b_0) + b_1^2} - b_1\right]; \; k \text{ positive integer}$$

Once the resample times are calculated, the amplitude of the signal at those times is calculated by interpolating between the sampled data using any

appropriate method. HP uses only linear interpolation; it will be shown that this simplification can cause inaccuracies. Other methods will be discussed in Section 4, Results & Discussion.

After the amplitudes are found, the resampled data are transformed from the angle domain to the order domain by means of a Fast Fourier Transform (FFT), complete with the application of data windows.

2.3.4 Bruel & Kjaer Method of Computed Order Tracking

The computational order tracking method proposed by B&K [20] uses the same inputs as the HP method but uses a different algorithm, based on high speed digital techniques called decimation and interpolation, to extract the speed-normalized data. Detail regarding the precise nature of the B&K method appeared too late to be incorporated in the simulation. Future development may add this method to the simulation and quantitatively compare it to others.

3 Equipment & Procedures

An original computer simulation was created to examine computed order tracking (COT) and compare it to other methods. By using a total simulation to test and compare the different methods, it was possible to ensure that only the factors of interest were changed from one test to the next. An apparatus for physical experimentation would not be expected to provide such repeatable raw data.

In addition, it is generally less expensive and time-consuming to alter simulation software than it would be to make similar changes to hardware. Another advantage of using a computer simulation is the ability to obtain the exactly correct results. Because all variables are known, a method that can find the exactly correct signal amplitude for a given point in time can be developed. The results from this exact method can be used as a base to compare all other results.

3.1 Equipment

The analyses done in this paper were performed using an original numerical simulation written using the C language and run on a personal computer. Elements in the simulation included the rotating machine, transducers, amplifier, analog-to-digital converter, and the processing algorithms. Both commercial and custom software were used to analyze and present results.

3.2 Use of Published Algorithms & Data

Some algorithms and data used in the simulation were created by others and used with only small changes. The details of these are discussed below.

3.2.1 Hewlett Packard Method of Computed Order Tracking

To create a fair and accurate representation of the HP method of computed order tracking, it was necessary to follow the original algorithm as closely as possible. Potter of HP had patented the method, so the patent [12] was obtained and used as a blueprint for writing that portion of the simulation. The code is shown in the appendix. Unfortunately, Potter did not include details of his linear interpolation filter because, as he explains,

> ... there are innumerable ways of interpolating digitized waveforms that are well known to those skilled in the interpolating art, [so] a specified interpolation process is not described here. Obviously, among other things the choice of a

> specific interpolation process to be used in an actual embodiment of the invention will depend upon economics and the level of interpolation sophistication necessary. [12]

Thus it was necessary to use methods of filtering and interpolating the data which may or may not be the same as the actual methods.

3.2.2 Digital Filter Coefficients

Where digital filters were used, the coefficients were calculated using commercial software called PCDSP [17]. This program allows the user to specify filter characteristics from which it generates an appropriate set of coefficients. This method allows better use of more complicated filter design techniques than manual calculation. The specifics of the digital filters are described in Section 4.3.2.1, Digital Filtering.

3.2.3 Block Cubic Interpolation

Two different methods of performing a cubic polynomial interpolation were tried. One method used a large block of raw data to generate spline coefficients [19], while the other used four raw data points at a time in a basic polynomial approach. The code for the four-points method was original. The spline method is described in more detail in Section 4.3.4, Effects of Interpolation Method.

3.2.4 Fast Fourier Transform

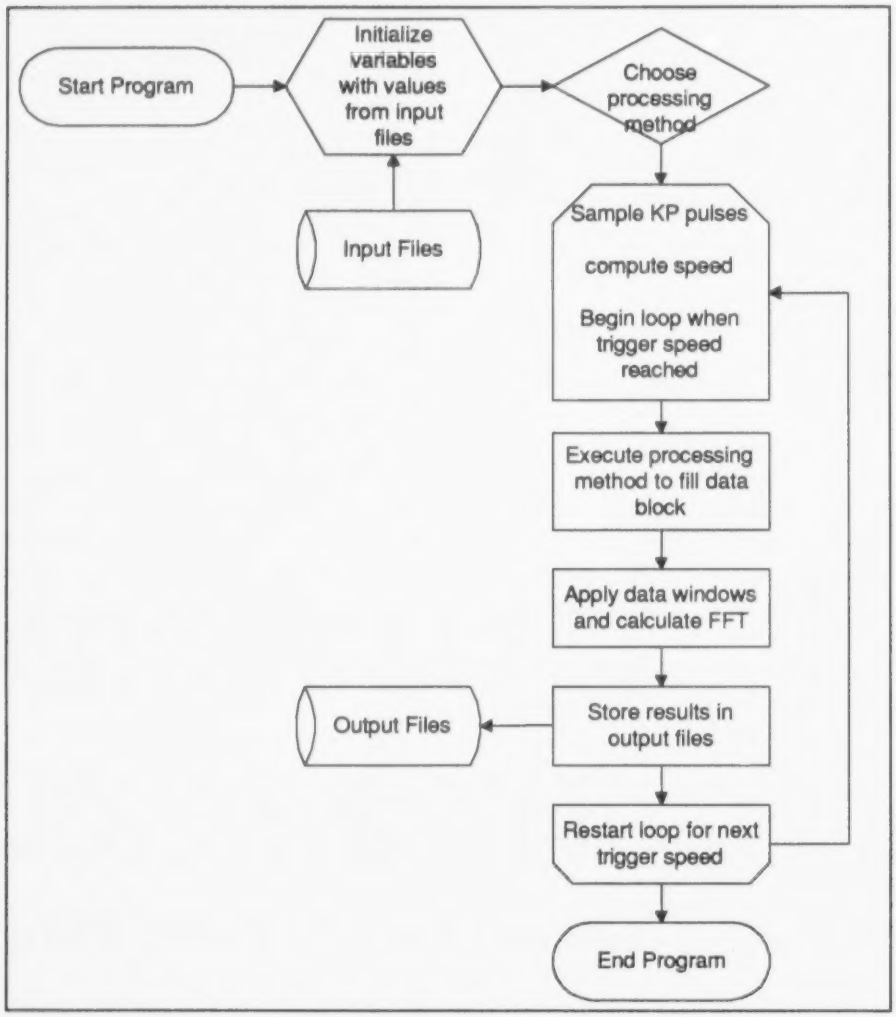
In the interests of speed, accuracy and reliability, published code for the Fast Fourier Transform was used for that portion of the simulation. This code was obtained from *Numerical Recipes in C* [18]. Minor modifications made the code compatible with the compiler used.

3.3 Procedure

In its simplest form, the simulation algorithm operates as illustrated in Figure 14. The code for this program is given in the appendix. Initialization (input) files controlled the simulation program to ensure that only the desired settings changed. Because all input and output are done with text files, the program results are almost self documenting, especially if related input and output files are stored in one location and given similar names.

For the majority of the tests, the reference signal consists of a vibration signal consisting of three components at orders 1.0, 2.5 and 4.0 and of amplitudes

1.0, 0.5 and 0.5, respectively (units of amplitude are arbitrary). The machine is defined to begin at rest and accelerate at a rate of 10 rpm/second with one keyphasor pulse per revolution. The sampling rate for the vibration signal is set at 250 Hz to avoid aliasing at 1000+ rpm, the highest analyzed shaft speed. The analog-to-digital converter (ADC) has double precision.

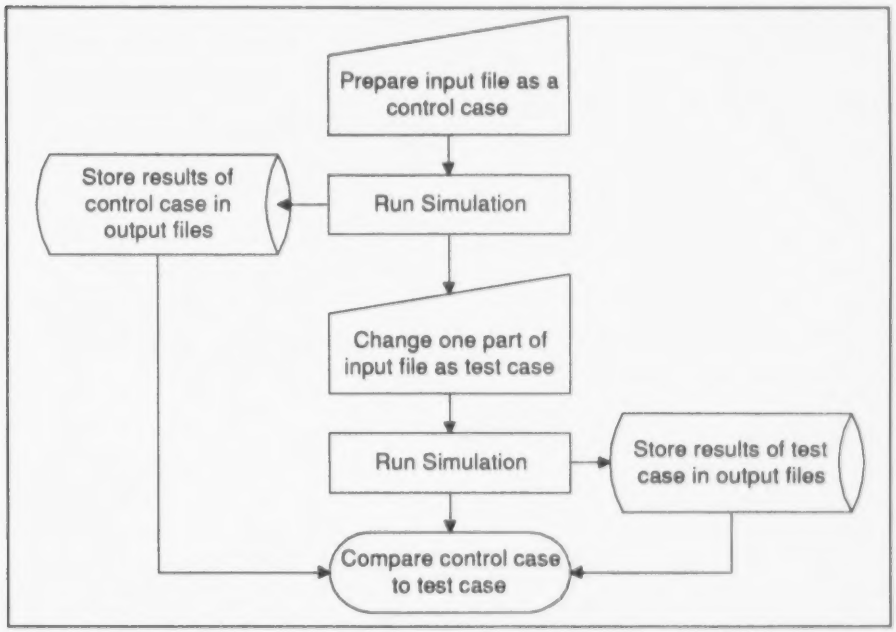

Start Program
Initialize variables with values from input files
Choose processing method
Input Files
Sample KP pulses
compute speed
Begin loop when trigger speed reached
Execute processing method to fill data block
Apply data windows and calculate FFT
Output Files
Store results in output files
Restart loop for next trigger speed
End Program

► Figure 14: Flowchart of simulation program

The simulation tracked the shaft speed of the machine by monitoring the time between consecutive keyphasor pulses. Sampling the vibration signal started once the shaft speed matched or exceeded the target speed. The process was suspended immediately after 512 data points were acquired and an FFT was

calculated on the block of data. The data block was chosen to be 512 points in size to achieve a compromise between storage requirements and processing time on one hand and frequency resolution on the other. The processing sequence resumed after the machine speed passed through the next target.

To make a test or comparison, the procedure illustrated in Figure 15 was used.


Prepare input file as a control case
Store results of control case in output files
Run Simulation
Change one part of input file as test case
Run Simulation
Store results of test case in output files
Compare control case to test case

► Figure 15: Flowchart of simulation procedure

4 Results & Discussion

Spectral information for machine run-ups is presented here using two-dimensional semilog plots. The vertical axis displays vibration magnitude on a log scale with arbitrary units and the horizontal axis displays orders.

4.1 Processing Methods

Three methods of order tracking analysis were modeled in the simulation: *exact*, *classical* and *computed*. The definition of these terms, as used in this study, are given below:

The *exact* method is an artificial but useful creation. Because all variables are known within the simulation, it is possible to create a method which samples the transducer signal at precisely the right times to obtain exactly constant $\Delta\theta$ samples. Spectra thus generated are used as a base for comparison (see Section 4.2, Similarity Factor).

The *classical* method is an approximation of the analog method of order tracking as described in Section 2.6.3, Traditional Means of Order Tracking. This method uses a variable sampling rate which is changed when each keyphasor arrives and is held constant until the next keyphasor arrives.

The *computed* method is a reconstruction of the computed order tracking method described in detail in Section 2.6.4, Hewlett Packard Method of Computed Order Tracking.

4.2 Similarity Factor

When discussing accuracy of the methods, it is necessary to compare the test results to a standard. This raises two questions: "How should the results be compared?" and "What should be used as the standard?"

Many critical machine components such as bearings and gears generate small vibration amplitudes. These usually occur in the presence of large amplitude vibrations caused by normal operation of the machine. For example, a fan might have a high amplitude peak at the blade passing frequency, and a significantly smaller peak from a critically worn bearing. Vibrations from the blade passing frequency are normal and acceptable, but the small vibration from the bearing is the information of interest. Thus, when comparing factors that affect accuracy, one must equally consider effects on both large and small amplitudes.

Berry [3] discusses the importance of the small sidebands present in rolling element bearing vibration signatures. The presence of rolling element bearing defect frequencies of any amplitude indicate a problem. If these appear, especially if accompanied by harmonics, or sidebands, the bearings should be replaced as soon as possible.

To compare two spectra, a *similarity factor* has been defined. This number can range from zero (completely dissimilar) to one (identical). It is computed by averaging the magnitude of fractional deviations of each data point in one spectrum from its counterpart in the other, and subtracting this average from unity, so:

$$S = 1 - \frac{1}{N} \sum_{1}^{N} \frac{l_i - s_i}{l_i}$$

where l_i is the larger and s_i is the smaller

This approach was chosen because it gives roughly the same weight to the small amplitudes as the large amplitudes in the spectra. In cases where the larger was exactly zero (and therefore the smaller is zero also, since the spectra contain only positive values) the two points were defined to be exactly similar.

The standard chosen for comparison is the spectrum produced by sampling the signal at exactly the right shaft angles (the "exact method") unless noted otherwise.

4.3 Comparison of Methods

Both the approximation of the classical method and the computed order tracking method yield acceptable results when compared to the exact method, as shown in Figure 16. Although the signal contains three frequencies, even the exact solution shows three wide triangular peaks, instead of three thin spectral lines. This occurs because the data block contains a non-integer number of shaft cycles (i.e., 512 samples / 12 samples per revolution = 42.7 revolutions). When this happens, the FFT adds extra frequencies to the spectrum because of a perceived discontinuity in the signal. Data windows (described in detail in Section 2.3.1, Fast Fourier Transform), which have been applied to all methods, reduce this effect but do not eliminate it completely.

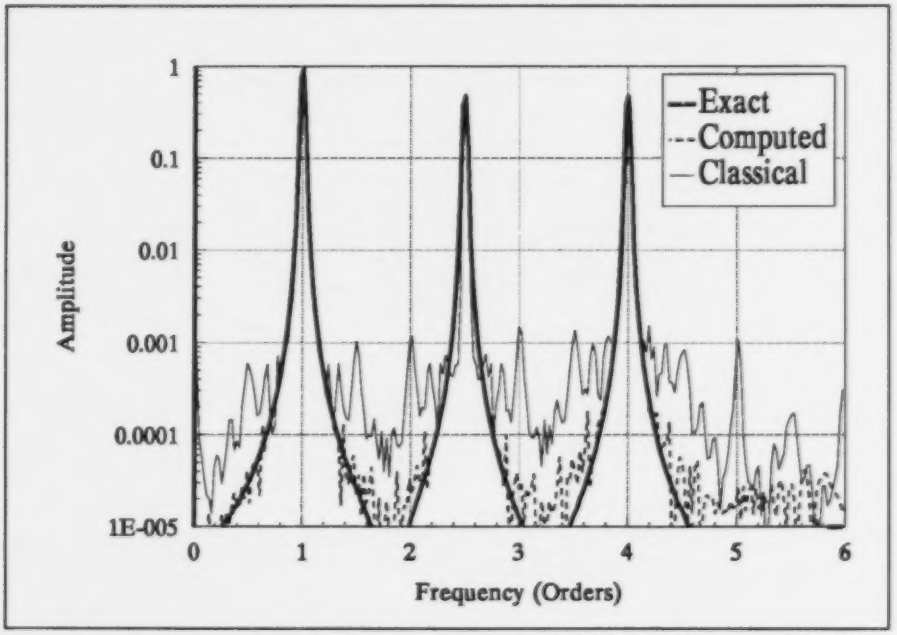
The answer to the question, "How well does COT work compared to the classical method?" depends on the quality of analog instrumentation used for the classical method (and higher quality is usually more expensive).

A major limitation of the classical method is reported to be tracking error in the ratio synthesizer, causing error in determination of rotation speed [13]. For example, if a ±5% error in determination of rotation speed is introduced to the classical method, a higher noise floor results. With all other parameters the same, the computed method produces cleaner spectra.

The approximation of the classical method assumes a linear increase in shaft angle with time, which results in occasional errors in the number of samples taken per revolution, causing some harmonic peaks to appear in the spectra.

In Figure 16, the three main peaks of the computed method are shaped more like the exact solution (similarity 0.551) than are the irregular peaks of the classical method (similarity 0.189). Also worth noting is the difference in the level and consistency of the noise floors. The noise floor in the computed method is much lower and flatter than the classical method.

The remainder of this study will focus on the HP method.



► Figure 16: Comparison of classical order tracking to other methods

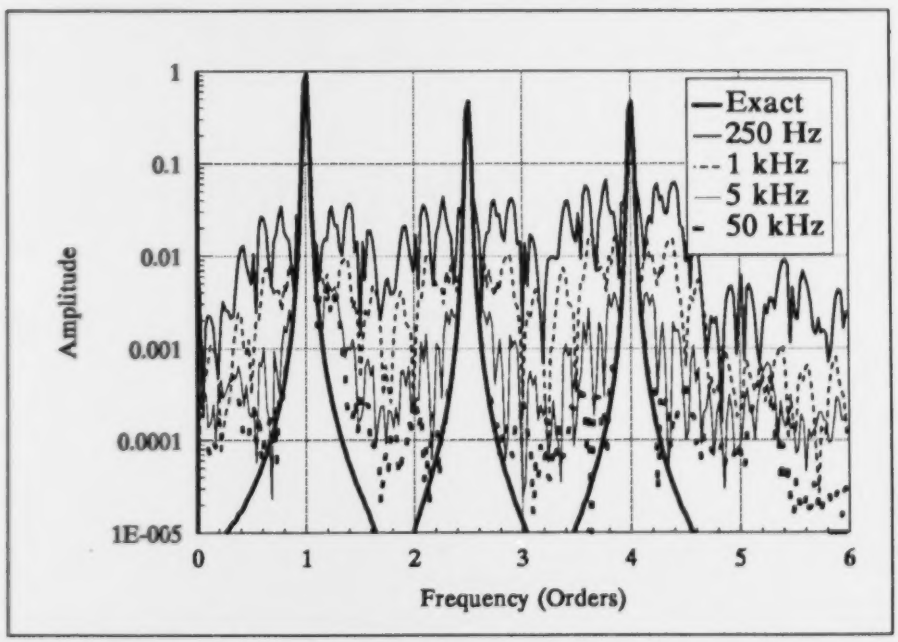
4.4 Factors Affecting Accuracy of the HP Method

4.4.1 Effects of the Keyphasor

4.4.1.1 Keyphasor Timing Accuracy & Resolution

The entire method of COT hinges on the accuracy of the resampling process. The resampling process is based on obtaining the times at which the keyphasor passes a certain point. The more accurate these times are, the more accurate the resampled data. The more accurate the resampled data, the more accurate the resulting spectra.

As illustrated in Figure 11, both the vibration signal and the keyphasor signal are discretely sampled. The rate at which the keyphasor signal is sampled determines the resolution of the keyphasor pulse arrival times. In other words, all keyphasor pulse arrival times will be integer multiples of the keyphasor channel sampling interval, Δt. The faster this sampling rate, the smaller this Δt becomes, resulting in better keyphasor timing resolution. This helps reduce error, as shown in Figure 17, where a number of different keyphasor sampling rates are compared. Figure 17 shows that an order of magnitude improvement in the keyphasor sampling resolution produces roughly an order of magnitude improvement in the noise level.



► Figure 17: Effects of keyphasor resolution

Special note should be made of the lowest keyphasor sampling rate, 250 Hz, the same sampling rate used for the vibration signal. Although this is an adequate sampling rate for the vibration signal (i.e. it is above the Nyquist limit), it is woefully inadequate for the keyphasor signal. The large multiple side peaks in the spectra would mask any bearing defect frequencies.

Common two channel data acquisition hardware often uses the same sampling rate for both channels. In that case, the choice of sampling rate must be determined by the requirements for keyphasor resolution. Although this increases the data storage requirements, it also improves the interpolation accuracy, discussed in Section 4.3.4, Effects of Interpolation Method.

4.4.1.2 Using Multiple Keyphasor Pulses per Revolution

Although one keyphasor per revolution is the most common, it is possible that more pulses per revolution may be available. (However, one thing must be remembered: computed order tracking assumes an equal angular separation between keyphasor pulses.) There is a practical limit on the number of keyphasor pulses per revolution. As mentioned in Section 2.6.4, Hewlett Packard Method of Computed Order Tracking, data between the most recent three keyphasors is used for resampling, and only those raw data in the center half of that time interval are used (to prevent overlap). Thus it is possible that if there are too many pulses, no raw data signals will fall into this interval.

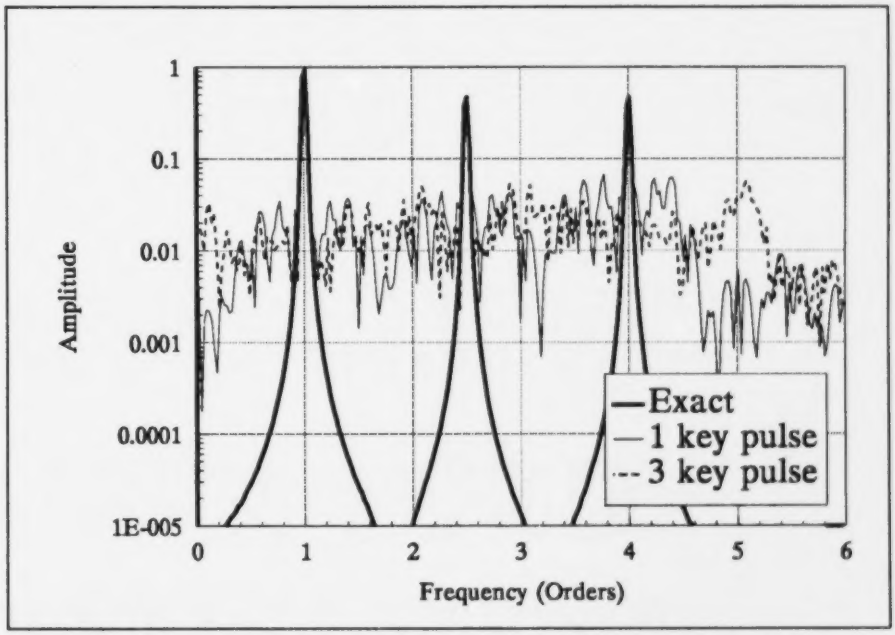
Furthermore, the three most recent keyphasor pulses span a certain range of shaft angle. If there are more keyphasor pulses than resamples per revolution, then no resample times may fall in the center half of the interval. The method could be refined to handle these circumstances, but the real need is questionable, in light of the following findings.

A low sampling rate (250Hz) for the keyphasor pulses was used in these tests to obtain a visible difference between the spectra. This low sampling rate is responsible for the high noise floors in Figures 18 and 19. When that difference emerged, it revealed that more keyphasor pulses per revolution are not necessarily a good thing. Table 1 details the difference between the spectra shown in Figures 18 and 19.

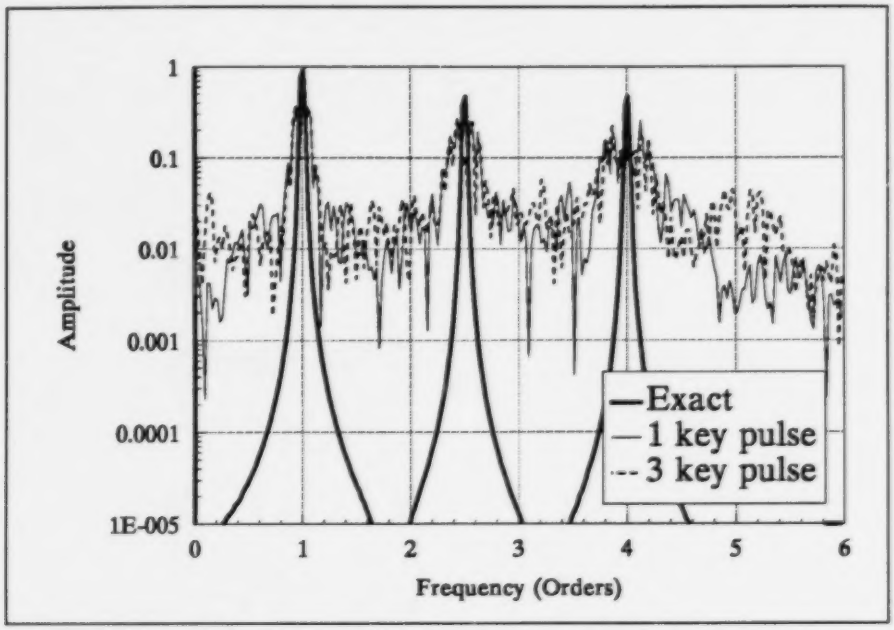
Table 1: Similarity comparison

	Similarity	
	constant acceleration	nonconstant acceleration
1 keyphasor	0.0721	0.0328
3 keyphasors	0.0767	0.0306

It is believed that more keyphasors cause higher distortion because any error in the determination of their arrival times is compounded by the (iterative) resampling calculations. These problems are exacerbated when the shaft speed changes in a nonlinear fashion. The effects of varying acceleration are discussed in Section 4.4.3.4, Varying Acceleration.



► Figure 18: Effect of using multiple keyphasors with linear acceleration



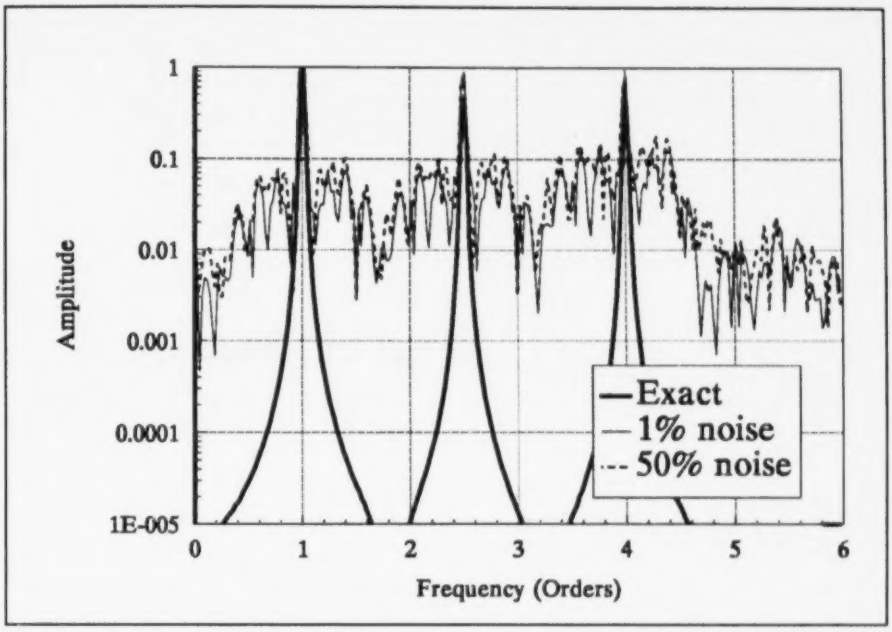
► Figure 19: Effect of using multiple keyphasors with nonlinear acceleration

4.4.1.3 Keyphasor Noise

In this test, the keyphasor signal was sampled at 250 Hz; when the keyphasor pulses arrived, a random error was added to the arrival times. Two tests were run, as shown in Figure 20. The first had low noise: the random error was between ±1% of Δt. The second had high noise: random error between ±50% of Δt. The spectral noise floor is generally high because of the low keyphasor signal sampling rate.

The results of these tests show a high noise floor (due mostly to the low keyphasor sampling rate) which increases only marginally when the keyphasor noise is increased to 50%.

It is somewhat surprising that the effects are not more drastic. In every other keyphasor test, any reduction of keyphasor precision resulted in a dramatic increase in the spectral noise floor. Perhaps the acceleration of the running speed was so low that the resample times happened to be very close to the correct placement. If the acceleration was greater or not constant, maybe the effects of random error in keyphasor timing would be more serious.



► Figure 20: Effect of adding random error to keyphasor signal

4.4.2 Effects of Filtering

4.4.2.1 Digital Filtering

When digitizing physical processes, the vibration signal is filtered by an analog lowpass filter to avoid aliasing and then is sampled at constant Δt. The required cutoff frequency for this filter is found by considering the resampling rate and the maximum rotation speed of the machine at which sampling will be done (i.e. the highest order at the highest speed).

Near this maximum speed, the sampling rate is very close to the resampling rate and so the analog filter is not only removing unwanted high frequencies, but also unwanted high orders. However, at lower speeds the vibrations occurring at the same unwanted high orders are not manifesting themselves as such high frequencies, so they are passed through the analog lowpass filter. These high orders must still be removed so they do not corrupt the spectra.

Since these high orders passed through the analog filter, they now exist in digital form within the sampled signal, and must be removed by some form of digital filtering. These orders have not been aliased, since they exist in the constant Δt signal. Should this signal be resampled without digital filtering,

these high orders will be aliased into the angle domain (the constant $\Delta\theta$ data) The data were sampled at equally spaced time intervals, so our digital filter must operate using frequencies, not orders, even though high orders are being removed. The cutoff frequency of the required digital filter is found by multiplying the cutoff order by the machine's speed. It is primarily the cutoff frequency that determines the filter coefficients.

If analyzing a machine running at constant speed, the cutoff frequency (and therefore the filter coefficients) would not change, and only one set would be needed. During a run-up and rundown, however, the cutoff frequency continually changes, requiring the filter coefficients to be different for every data point. One set of coefficients may be used for each trigger speed if the machine is not accelerating too fast, but this may lead to some attenuation of desired high orders (i.e. those just below the correct cutoff frequency).

4.4.2.1.1 Filter Design

All the spectra in this study use a frequency range of 0 to 6 orders, obtained by resampling the data 12 times per revolution. Thus, the desired cutoff frequency, f_c, is 6 orders. At 100 rpm, the time based equivalent of 6 orders is

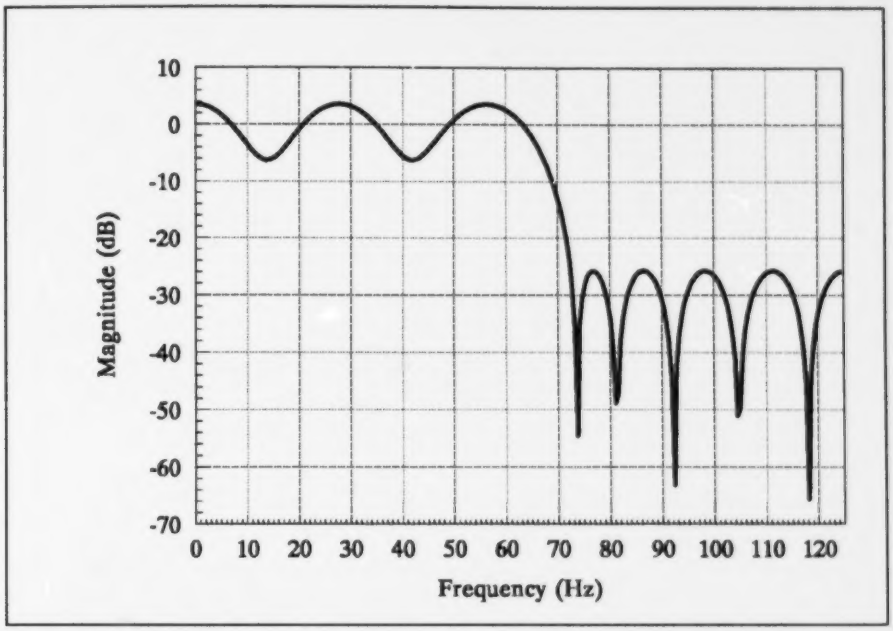
$$f_c = 6\frac{\text{cycles}}{\text{rev}} \times 100\frac{\text{rev}}{\text{min}} \times \frac{1\text{ min}}{60\text{ sec}} = 10\,\text{Hz}$$

So for 100 rpm, the cutoff frequency is 10 Hz and the sampling rate is 250 Hz. Thus, the cutoff for the initial filter is ideally

$$f_c' = \frac{10\,\text{Hz}}{250\,\text{Hz}} = 0.04 f_c$$

The cutoff frequencies for filters at other trigger speeds were found in the same fashion. After trying different methods, a two band Parks-McClellan design method was chosen to generate 21 coefficients for a nonrecursive finite impulse response filter. The response for the 700 rpm filter is shown in Figure 21.

Obviously, this filter is not ideal. If a skilled filter designer were to apply their efforts, a much improved filter shape would result. Improvements to these filters would include a reduction in the pass band ripple and an increase in the difference between the pass band amplitude and stop band amplitude (attenuation). However, this shape and those for other trigger speeds will suffice for the purposes of this study.



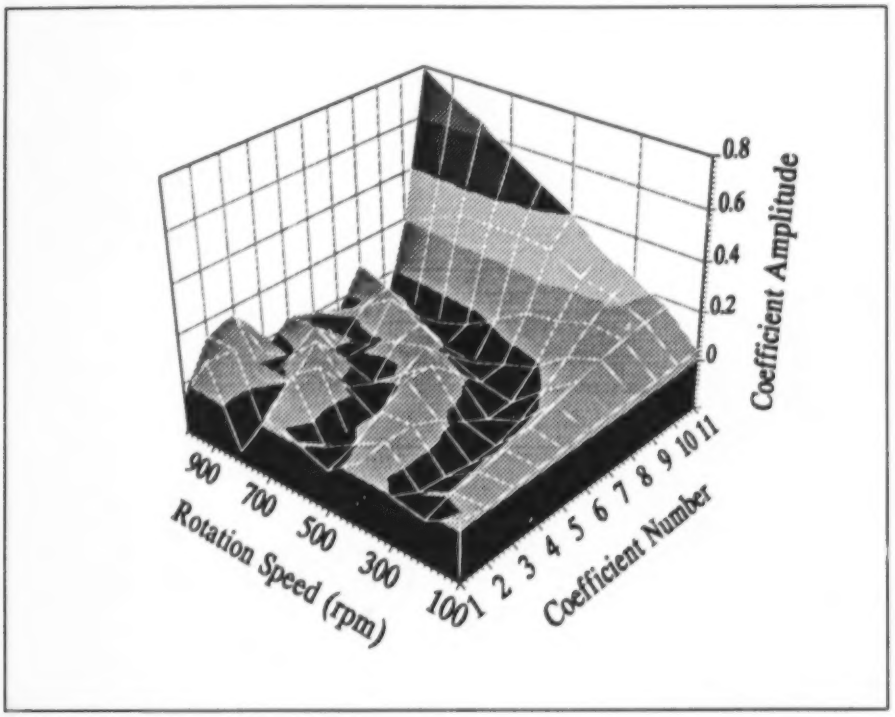
► Figure 21: Digital filter frequency response for 700 rpm trigger speed

4.4.2.2 Interpolated Digital Filtering

When analyzing a machine running at constant speed, the cutoff frequency would not change, and only one set of filter coefficients would be required. During a run-up and rundown, however, the cutoff frequency continually changes, requiring the filter coefficients to be different for every data point.

Defining a set of digital filter coefficients for every data point is not a practical approach, but it can be approximated by interpolating a new set of filter coefficients from the existing sets. Figure 22 shows that the sets of filter coefficients form a family of curves. Knowing the arrival times of the keyphasor pulses, we can find the precise machine speed at each data point and interpolate an appropriate set of filter coefficients for that data point. This may be the approach used by HP, as indicated by this discussion of their interpolation filter,

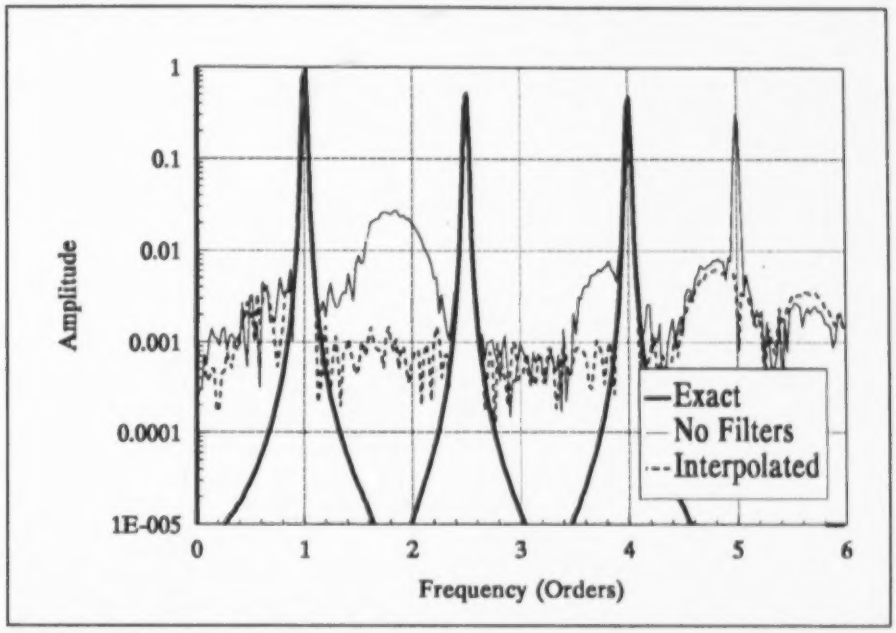
> To improve speed, the actual filter is implemented as a look-up table in memory, which leads to some round-off errors yielding the desired 80 dB dynamic range. [7]



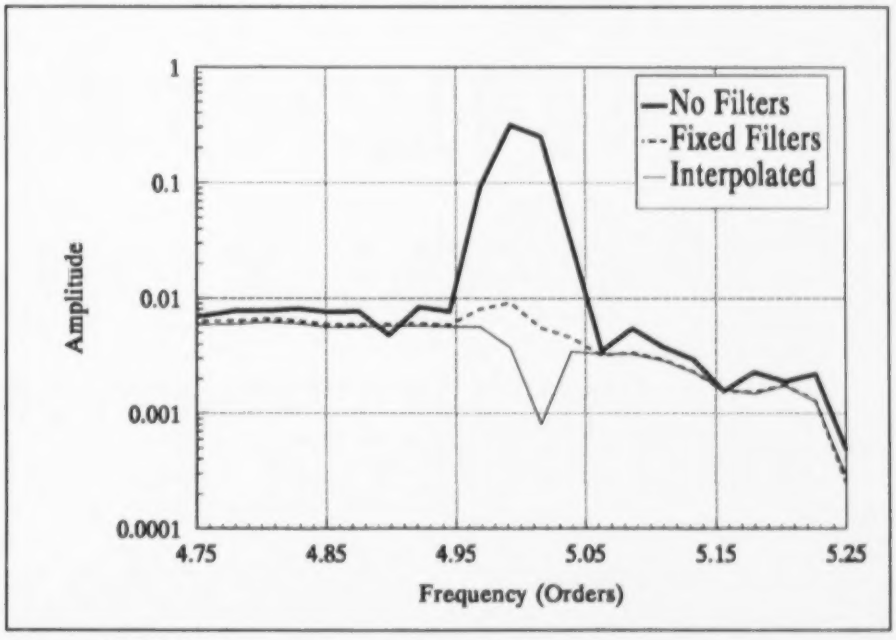
► Figure 22: Family of digital filter coefficient curves

To test the effects of using digital filtering when resampling the data, a vibration has been added at 7 orders with a 0.5 amplitude. This frequency of this vibration component is too high for the resampling rate; without digital filtering, this component is expected to be aliased or folded back around 6 orders and appear as a peak at 5 orders. The expected effect does occur, as shown in Figure 23. Digital filtering is implemented to remove such data and produce a spectrum as if the 7 order vibration had never been included in the signal. Random noise in the amount of 1% has also been added to this signal to better model a real world situation.

Figure 24 shows a close-up of the aliased peak at 5 orders. The peak is removed by both the fixed filter and interpolated filter methods (which were tried independently), but the interpolated method produces consistently lower noise amplitudes. The similarity factors for the interpolated filter method is 0.178, as compared to 0.171 for the fixed filter method, an improvement of 4.3%.



► Figure 23: Removal of high orders using digital filtering



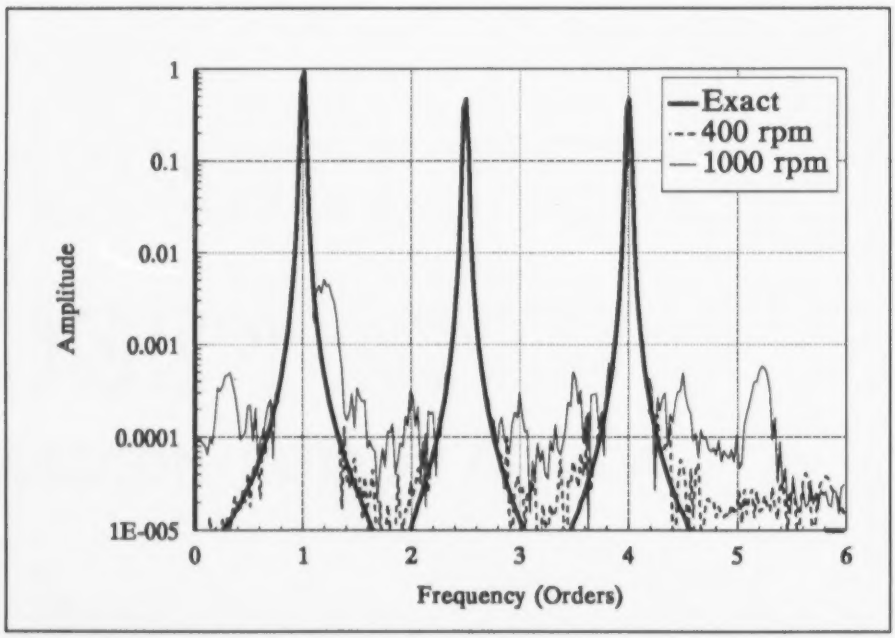
► Figure 24: Detailed examination of filter effects

4.4.3 Effects of Rotation Speed & Acceleration

Tan and Mathew [16] report on some limitations of order tracking analysis relating to shaft speed. They mention that "the effectiveness of order tracking analysis decreases when the rotational speed changes rapidly". Because they make references to "conventional order tracking instrumentation", they are probably discussing limitations of the classical method, not computed order tracking. This is not to say that COT is immune to rapid changes in rotation speed, but it seems to be less affected than the classical method.

4.4.3.1 Assumption of Consistent Shaft Angle

Like the vibration signal, the keyphasor signal is sampled at constant time intervals. This gives fixed precision in the measurement of the keyphasor pulse arrival times. In the resampling algorithm, a certain shaft angle (zero radians for one keyphasor) is assumed at the keyphasor pulse arrival time. As the machine accelerates, this assumption becomes less valid. The effects are illustrated in Figure 25.



► Figure 25: Increased rotation speed decreases keyphasor accuracy

Physically, the keyphasor passes the sensor, causing it to generate a signal. By the time the acquisition hardware recognizes, samples the signal and records an

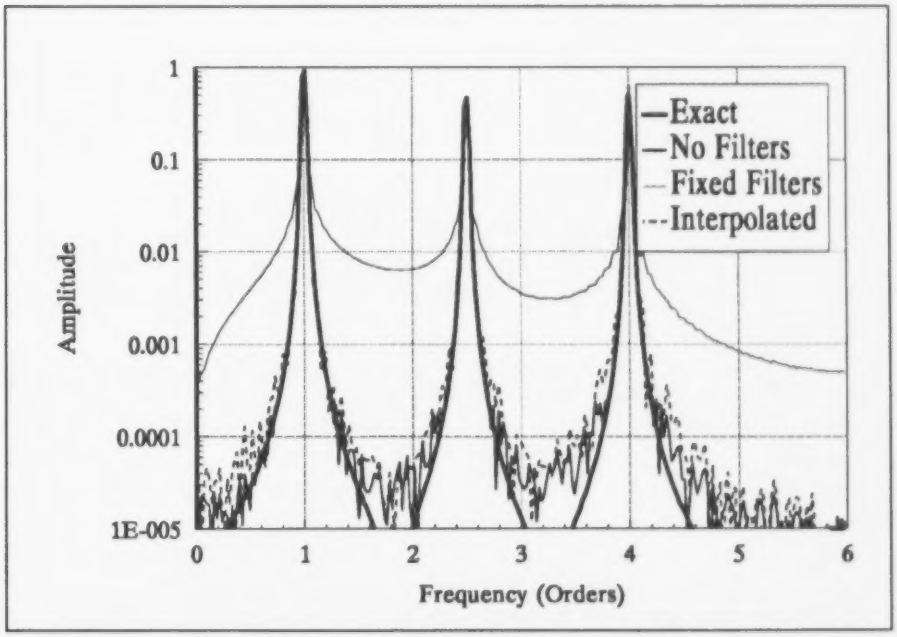
arrival time, the keyphasor has moved to a different angle. Thus, the keyphasor timing precision continually becomes less valid.

The result is a decreased accuracy in measuring the keyphasor pulse arrival times. As discussed earlier in Section 4.4.1, Effects of the Keyphasor, this results in an overall decrease in spectral accuracy. The spectra begin with low noise levels for low trigger speeds and end with high noise levels at higher trigger speeds, as shown in Figure 25.

4.4.3.2 Misrepresentation of Frequency Content

Generating a spectrum by means of an FFT requires a fixed number of data samples (this study used 512). Obtaining data at constant $\Delta\theta$ increments to obtain a fixed number of samples requires that the machine execute a fixed number of revolutions, regardless of the trigger speed.

If the rotation acceleration is too extreme, the last samples in the set are taken at a significantly different speed from the first ones. Thus, the spectra obtained may not be an accurate snapshot of vibrations present at the trigger speed. Notice the fixed filter curve in Figure 26; using a filter designed for 400 rpm on a signal which accelerates to nearly 500 rpm severely distorts the spectrum.



► Figure 26: Spectrum at 400 rpm during high acceleration

Taking this one step further, if the rotation acceleration is even more extreme, the next trigger speed may be reached before enough samples have been taken for the spectrum of the last trigger speed. This will cause large problems besides those already mentioned; since the method does not store the used raw data, it cannot go backwards to start from the missed trigger speed. The method could be modified to allow for this occurrence.

4.4.3.3 Attenuation of High Orders

If rotation speed is significantly different at the end of a data block as compared to the beginning, and if only one fixed-frequency digital filter is used for each data block, then the desired high orders contained in the last samples of the block will be removed. Over the length of the data block, the contribution from these high orders decreases as rotation speed increases and the signal outgrows the fixed-frequency filter. The resulting spectra do not show an absence of these high orders, but an attenuation of them; the resulting spectra will show the amplitude of those high orders to be lower than they really are.

Although this phenomenon can be predicted and envisioned, an example showing the occurrence could not be produced. The parameters of the test would probably have to be changed so drastically that the resulting spectrum would bear no resemblance to the comparison base.

4.4.3.4 Varying Acceleration

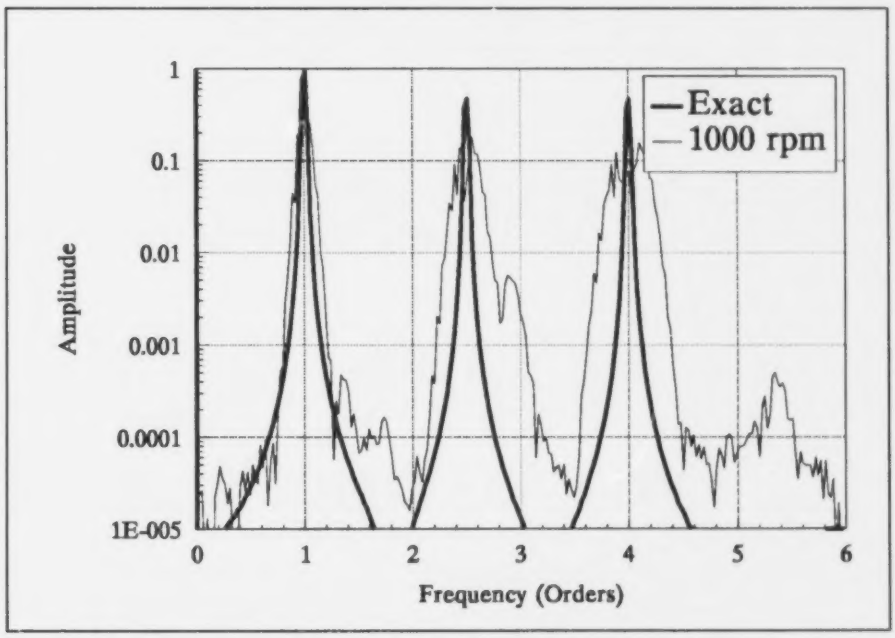
The HP method assumes a constant acceleration between keyphasor pulses. When this assumption is not valid, the interpolation times are less accurate. This causes two problems. First, the resamples are not taken at constant $\Delta\theta$, so peaks in the spectra may not occur at the right orders. Secondly, when the resample times are in error, the resample amplitudes will be inconsistent, causing a higher spectral noise floor.

Because constant acceleration does not occur in the real world, two things were done to intentionally violate the assumption. The first was to introduce a linear acceleration change over the course of the runup. However, the resample times are determined by fitting a quadratic curve to the keyphasor pulse arrival times to obtain a curve of shaft angle vs. time. Within the time span of three keyphasor pulses, this assumption provides good results even if the acceleration is changing in a linear fashion. With the rate of change kept realistic, the results were virtually unaffected and so they are not shown.

The second change was to introduce a moderate sinusoidal fluctuation in the rotation speed, intended to represent a varying change in acceleration. The results of this effect are discussed below.

The spectrum resulting from this nonlinear acceleration shown in Figure 27 has a similarity of 0.168. Notice the wide, irregularly shaped peaks and higher noise floor, all resulting from violation of the constant acceleration assumption. All tests were done with a very fast (50 kHz) keyphasor sample rate. When the keyphasor sample rate is lowered, extreme deterioration of the spectra can be expected, as discussed in Section 4.4.1.1, Effects of Keyphasor Timing Accuracy and Resolution.

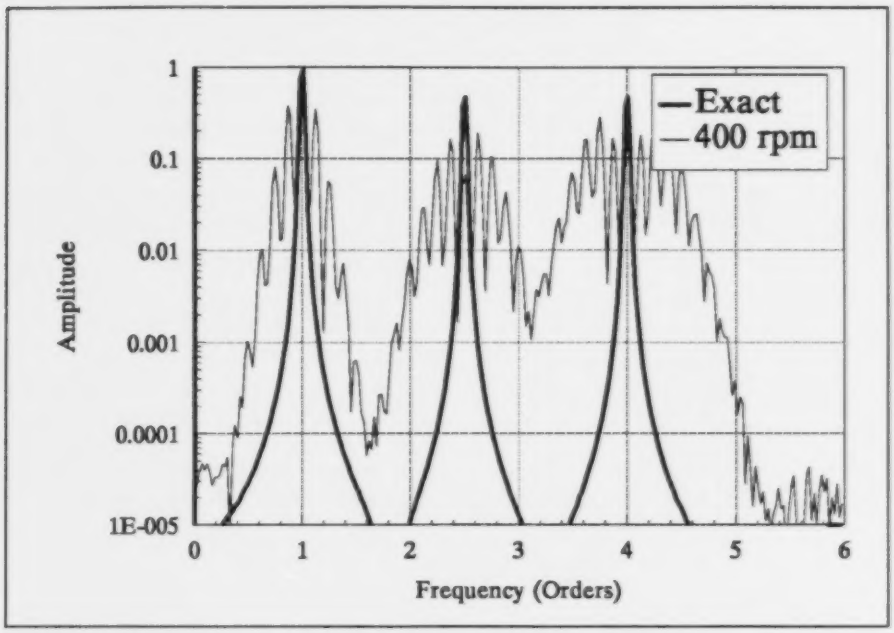
At lower rotational speeds, the results are even more dramatic, as shown in Figure 28. Because the time interval between successive keyphasor pulses is longer at lower speeds, the same fractional error in the resample times manifests itself as a larger absolute time error, result resulting in wider, more irregular peaks. These will hide or imitate bearing defect frequency patterns, especially where smaller sidebands are expected to appear.



► Figure 27: Spectrum at 1000 rpm during nonlinear acceleration

It is possible that using more keyphasor pulses per revolution would improve the method's accuracy during varying acceleration. However, as discussed

earlier (in Section 4.4.1.2. Using Multiple Keyphasor Pulses per Revolution), more keyphasors can also mean more noise in the spectrum.



► Figure 28: Spectrum at 400 rpm during nonlinear acceleration

4.4.4 Effects of Interpolation Method

Just as the accuracy of the keyphasor pulse arrival time determines the accuracy of the resample time, the accuracy of the interpolation method determines the accuracy of the resample amplitude. Typically, the best results for the least computational effort are desired to make the most of available computing resources. First, second and third order polynomial interpolation schemes were tried; however, it must be remembered that the vibration signals are produced by cyclical phenomena, and thus by nature the real function is sinusoidal, not polynomial. Therefore, these interpolation methods inherently incorporate error in the method. Higher order polynomial interpolations were not tried, as they were thought to be dissimilar to the inherent sinusoidal functions.

It will be shown that even at the minimum sampling rate, cubic interpolation can be very accurate. In fact, HP claims high accuracy using a *linear interpolation* filter.

Sinusoidal interpolation approaches were not tried because they require prior knowledge of all the frequencies in the signal. Regression techniques were not tried because the interpolation curve must pass through the data points (it is assumed that, after filtering, the original data points are no longer approximate), and a regression curve would not.

Four polynomial interpolation methods were tried. The first three (linear, quadratic, piecewise cubic) fit unique polynomials to the minimum required data points around the interpolation point. The fourth method (blockwise cubic) uses all available raw data points to ensure that the first and second derivatives of the interpolated curve are continuous over the entire data set, thus creating the smoothest possible cubic interpolation curve through the data.

These interpolation schemes all make the fundamental assumption that the data were produced by a smooth function; interpolation across a discontinuity rarely produces an accurate value. In reality this is a good assumption, since machinery vibrations are caused by the physical movement of objects, the paths of which cannot contain discontinuities. Shaft defects such as cracks and scratches can produce extremely sharp changes in the vibration signal which may look like discontinuities but are truly continuous. Furthermore, such rapid changes represent a combination of very high frequencies, and some of these are removed by the lowpass filtering (the filtering process helps to smooth out the signal before the data are even sampled).

At the extreme end of the test run, data are acquired starting at 1000 rpm and ending at a higher speed after 512 data points are acquired. At 12 points per revolution, this means roughly 50 shaft revolutions elapse after the machine reaches 1000 rpm. To push the limits of these interpolations, the sampling rate in the interpolation tests was lowered to 210 Hz.

Similar analyses were performed for each interpolation method discussed below. The first figure in each section (Figures 29, 31, 33 and 35) shows the actual results of an interpolation across the same data points. The second figure in each section (Figures 30, 32, 34 and 36) shows the spectra obtained at the start and end of each runup compared to each other and to the exact solution.

For each interpolation method, the spectra obtained at 1000 rpm contain false peaks. Since these peaks disappear when a higher sampling rate is used, they are known to be caused by the low (210 Hz) sample rate. The resulting interpolation error manifests itself as high orders, which alias about 6 orders and fold back into the spectra.

It is important to remember that greater accuracy is obtained when the sampling rate is high compared to the resampling rate. The interpolation tends to become less exact when the sampling rate and the resampling rate are close to the same.

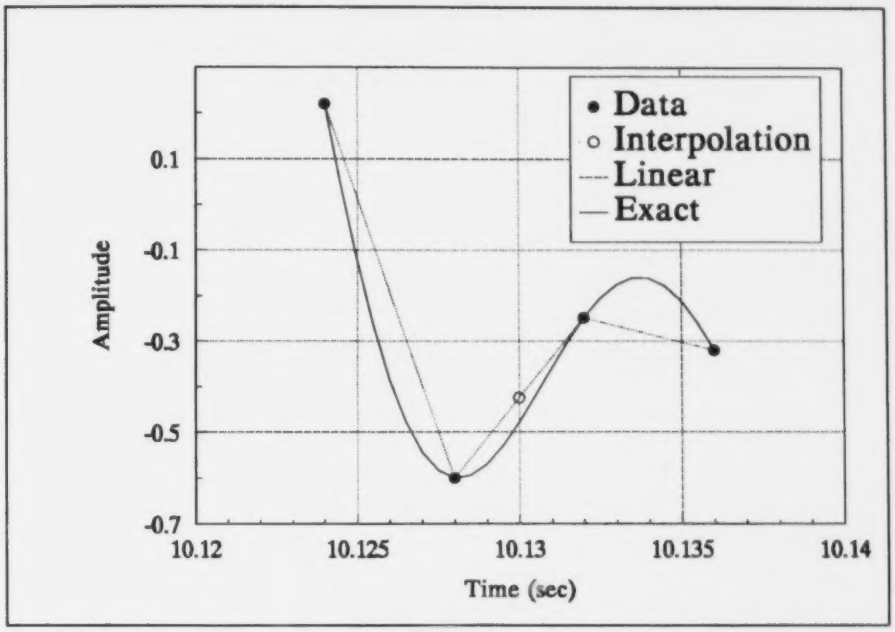
For example, if the sample rate is such that 120 samples are taken per revolution and the resample rate is 12 samples per revolution, then there will be 5 data points on each side of every resample point. Because the resamples are equally spaced, this means that those raw samples are more closely packed around the resamples. This will tend to give better results than if the there were 12 samples per revolution, where there would be only 1 data points on each side of (and further away from) every resample point.

4.4.4.1 Linear Interpolation

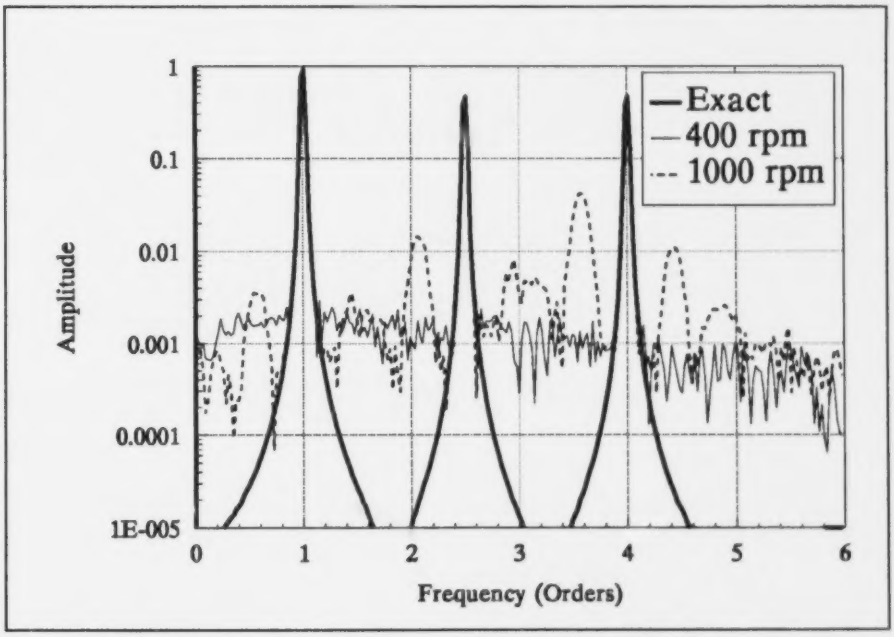
Because HP uses a linear interpolation filter for resampling the data, linear interpolation was the first method tried. This technique uses the raw data points on either side of the interpolation point, and then calculates the amplitude of the interpolation point using a lever rule. Due to the simplicity of the calculations, this is the only method tried for which double precision calculation is not strictly required.

When the transducer data are highly oversampled, (e.g., low speeds at the beginning of the run-up) linear interpolation is accurate because the data are so close together. As the sampling rate approaches the Nyquist criterion, (e.g., high speeds at the end of a run-up) the data are not as highly oversampled. In the example illustrated by Figure 29 (at roughly 1000 rpm, near the Nyquist limit), linear interpolation yields a value of -0.425, which is 11.1% greater than -0.478, the exact value. In these cases, higher order polynomial interpolation routines tend to provide better results because they approximate the shape of the real curve from which the data originated.

HP recommends using a sampling rate at least twice that suggested by the Nyquist criterion. The sampling rate at 400 rpm meets this recommendation; the sampling rate at 1000 rpm does not — it is essentially at the Nyquist limit. Notice the improved spectrum at 400 rpm in Figure 30.



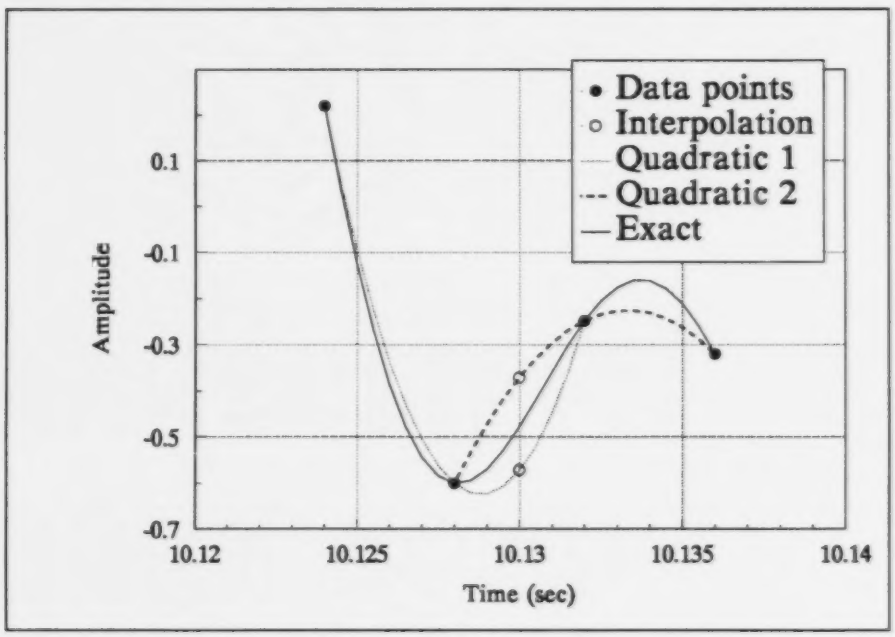
► Figure 29: Detail of linear interpolation near 1000 rpm



► Figure 30: Spectrum from linear interpolation at low and high speeds

4.4.4.2 Quadratic Interpolation

The method used here for performing quadratic interpolation is perhaps unusual. For a polynomial interpolation scheme using the minimum number of points, polynomials of odd orders allow equal numbers of original data points on either side of the interpolation point. Polynomials of even orders (such as quadratic interpolation) do not. For a quadratic interpolation, should the two data points before the interpolation point and the one after be used, or vice versa? Both ways give different values for the amplitude of the interpolated data point. Figure 30 illustrates the same example used for the discussion of linear interpolation, showing that the decision is not trivial. The exact amplitude, -0.478, lies between -0.392 and -0.589, the two quadratic results.

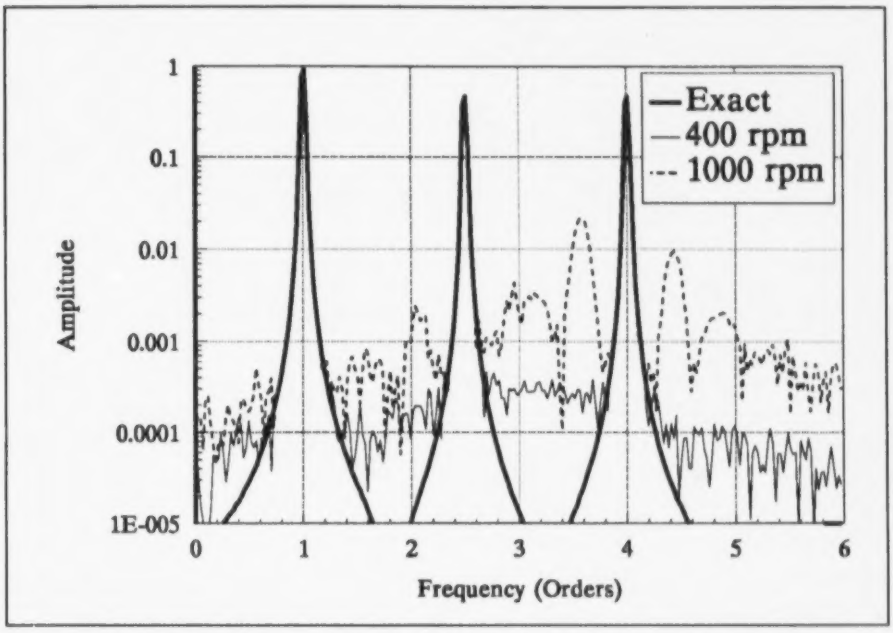


► Figure 31: Detail of quadratic interpolation near 1000 rpm

To achieve a final result, three steps are taken. First, a quadratic curve is fitted to the two data points before the interpolation time and the one after. A second quadratic curve is fitted to the one data point before and the two after. Finally, the amplitudes of the resulting interpolations are averaged to produce the final result. In the above case, this method results in a final result of -0.490, which is only 2.5% less than -0.478, the exact answer.

In Figure 32, the 400 rpm spectrum has a similarity of 0.36 while the similarity for the 1000 rpm spectrum is 0.24, a 33% deterioration over the course of the

runup. This method of quadratic interpolation yields spectra very close (similarity differs by less than 10%) to the piecewise cubic interpolation method, discussed next. This is not surprising, since both methods use the same data points.



► Figure 32: Spectrum from quadratic interpolation at low and high speeds

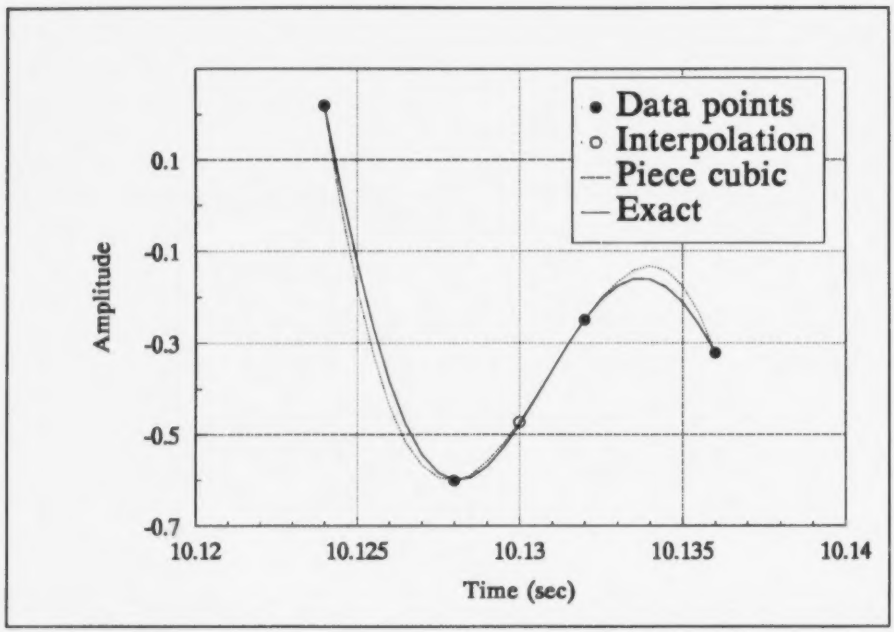
4.4.4.3 Piecewise Cubic Interpolation

The simplest method of programming a cubic interpolation is to use two raw data points before the interpolation point and two data points after, and fit a cubic curve to those four points. In other words, we start with the equation for the interpolating polynomial

$$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3$$

and four data points (i.e. pairs of x_i and y_i), with which we generate four independent equations. Standard techniques were used to solve this system for the coefficients $\{a\}$, which were used to find y (the signal amplitude) at any given x (resample time). Optimized techniques for these calculations have been published [18], but in the interests of coding simplicity, the standard solutions were implemented in the simulation program.

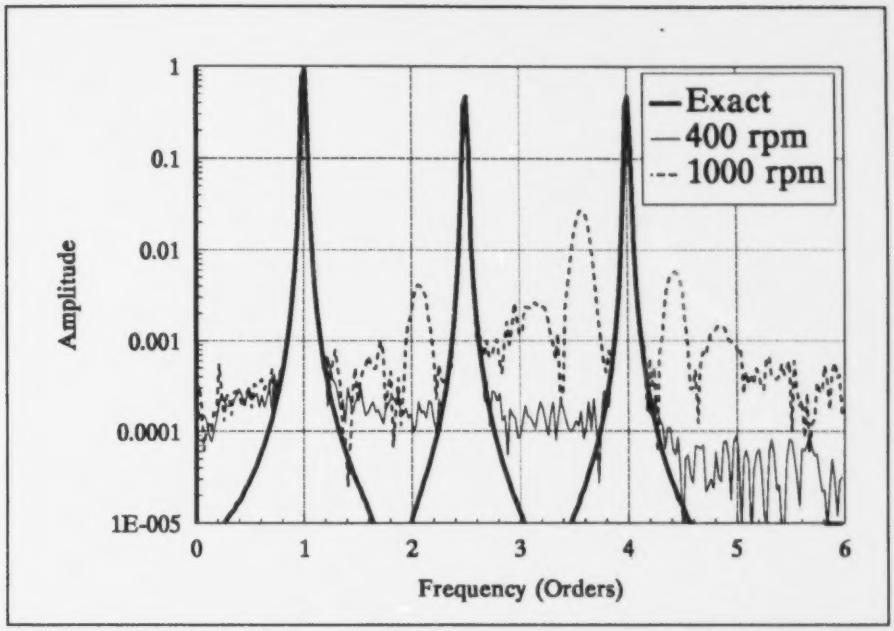
As shown in Figure 33, use of the same data points as the linear interpolation example yields an interpolated result of -0.473, compared to -0.478, the exact value. This is a difference of only -1.2%.



► Figure 33: Detail of piecewise cubic interpolation near 1000 rpm

This piecewise use of the data is extremely sensitive to the precision of the calculated coefficients, and accuracy suffers accordingly. In the next section, a blockwise cubic spline method is employed to overcome this problem.

In figure 34, the 400 rpm spectrum has a similarity of 0.33 while the similarity for the 1000 rpm spectrum is 0.24, a 27% decrease. Notice the equivalence of the quadratic and piece cubic interpolation method; at 1000 rpm, the similarity factor is 0.24 for both.



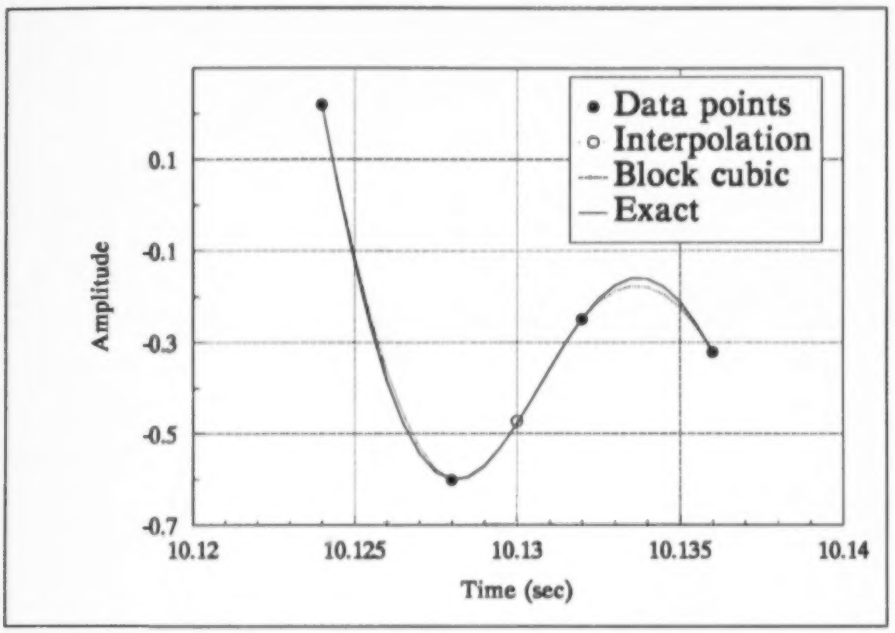
► Figure 34: Piecewise cubic interpolation spectrum at low and high speeds

4.4.4.4 Blockwise Cubic Spline Interpolation

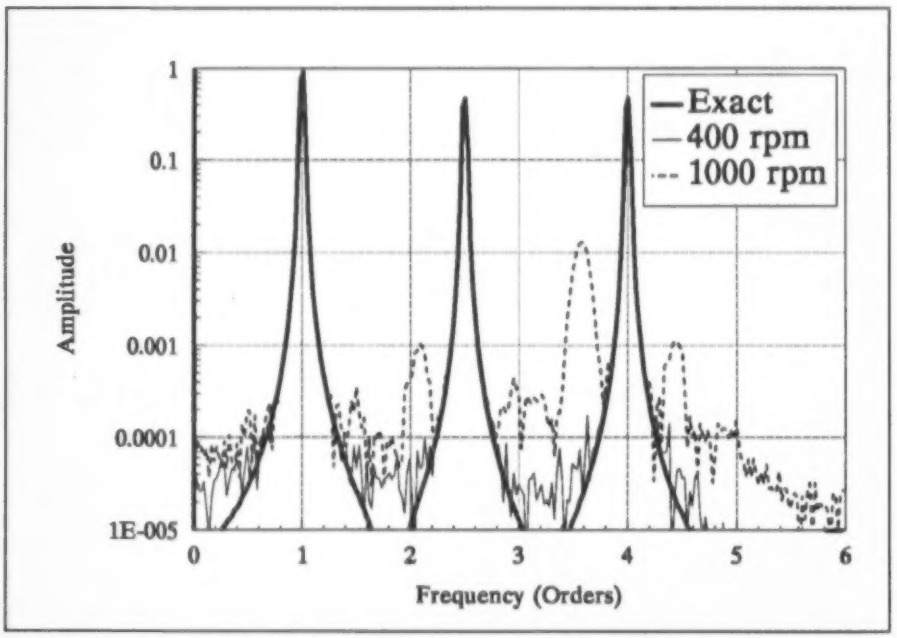
A more sophisticated method of cubic interpolation considers a larger block of raw data (in this method, all raw data between t_1 and t_3) and fits a series of cubic splines to that data [19]. Any interpolation points within that data block can then be found without having to recalculate the splines.

This routine generally produces better results than the piecewise cubic approach because the first and second derivatives of the interpolated curve are continuous, producing a smooth curve through all the data points. In the example illustrated in Figure 35, the interpolated value is -0.4778, which is less than 0.1% greater than -0.4782, the exact value. Notice how closely the interpolated curve follows the actual signal.

In Figure 36, the 400 rpm spectrum has a similarity of 0.56 (the highest yet) while the similarity for the 1000 rpm spectrum is 0.32 (the highest at 1000 rpm). This represents a 43% decrease in similarity over the course of the runup.



► Figure 35: Detail of blockwise cubic interpolation near 1000 rpm



► Figure 36: Spectrum from block cubic interpolation at low and high speeds

4.4.4.5 Interpolation Summary

In the interpolation example, all methods produced an interpolated value different from the exact value, as detailed in Table 2. This was expected because the true function is sinusoidal, not polynomial. Within this analysis, the higher order polynomials produce more accurate results. Of the two cubic methods, the block cubic method is more accurate than the piecewise cubic method, also as expected because of the smoother curve generated by the block cubic method and the reduced sensitivity to precision in calculation.

Table 2: Amplitude Comparison of Interpolation Methods

Interpolation Method	Interpolated Value	Exact Value	Error (%)
Linear	0.4247	0.4782	-11.19
Quadratic	0.4900	0.4782	2.47
Piece Cubic	0.4727	0.4782	-1.15
Block Cubic	0.4778	0.4782	-0.08

Table 3 quantitatively compares Figures 30, 32, 34 and 36. Although the block cubic routine shows the highest deterioration of similarity, -42.86%, it remains the most similar (similarity 0.32) spectrum at 1000 rpm. Although the linear method seems to improve over the course of the runup, it suffers from chronically poor similarity (similarity 0.16 to 0.18). Visual examination of the linear interpolation spectra in Figure 30 shows that any improvement exists in the similarity factor only. Notice the equivalence of the quadratic and piece cubic interpolation methods: at 1000 rpm, the similarity factor is 0.24 for both.

Table 3: Spectral Comparison of Interpolation Methods

Interpolation Method	Similarity		Change (%)
	400 rpm	1000 rpm	
Linear	0.16	0.18	12.50
Quadratic	0.36	0.24	-33.33
Piece Cubic	0.33	0.24	-27.27
Block Cubic	0.56	0.32	-42.86

Precision in calculation is critical for all but the linear interpolation method. It is important to recognize that in practice, the raw data have only a few reliable significant figures. Thus, single precision is normally sufficient for storage of the data. Double precision, when required, is used for *calculation* of the

interpolation (coefficients, etc.). The quadratic and cubic approaches require double precision for variables (roughly 15 significant digits); single precision (roughly 7 significant digits) is inadequate. The linear interpolation method is relatively insensitive to this change. When conserving computing resources, this is a strong argument for using linear interpolation in preference to the higher-order methods.

4.4.5 Considerations Regarding Noise

Noise is any component of an electronic signal which does not represent the variable intended to be measured [2]. Random noise in the time signal results in lower dynamic range and also raises the noise floor in the spectrum, masking any orders with such low magnitudes. As discussed previously, orders with low magnitude can be very important. To reduce noise, its source must be considered; noise from some sources can be reduced, while noise from others cannot.

Error in the interpolation of resampled data will generate more noise at higher machine speeds because the data points are further apart in the angle domain. Recognition of this fact can lead to minimum sample rate guidelines to keep interpolation accurate. Also, it may be possible to use a different, more capable interpolation routine that is less sensitive to high speed.

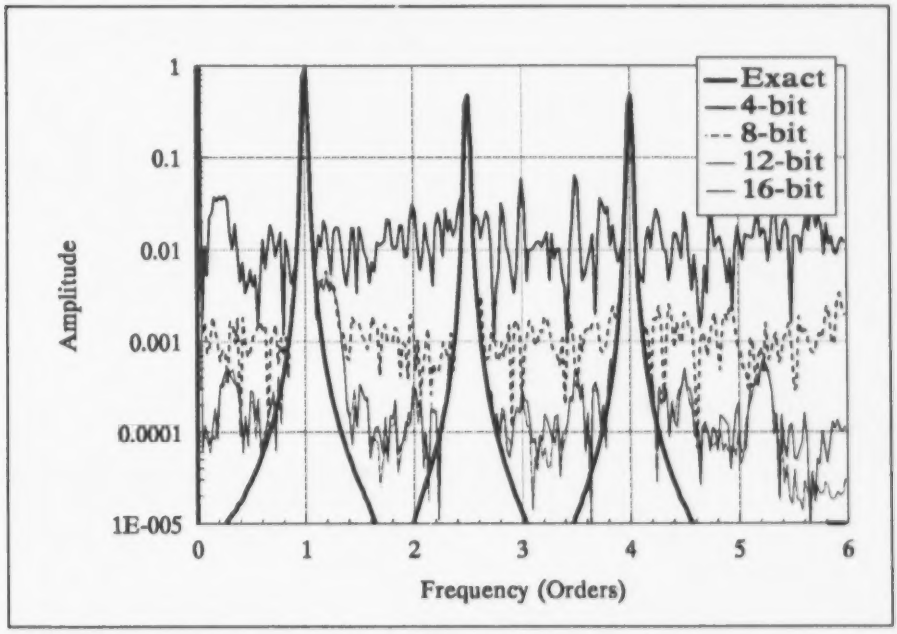
4.4.5.1 Sources of Noise

There are numerous possible sources of noise, but they can be divided into two types: internal and external. External noise can come from stray electrical sources, poor ground connections, faulty transducer mounting, bad calibration and analog-to-digital (ADC) quantization and word length. Internal noise can result from such things as roundoff error in calculations due to the available precision of variables and processor word length and error in calculation due to the calculation method chosen.

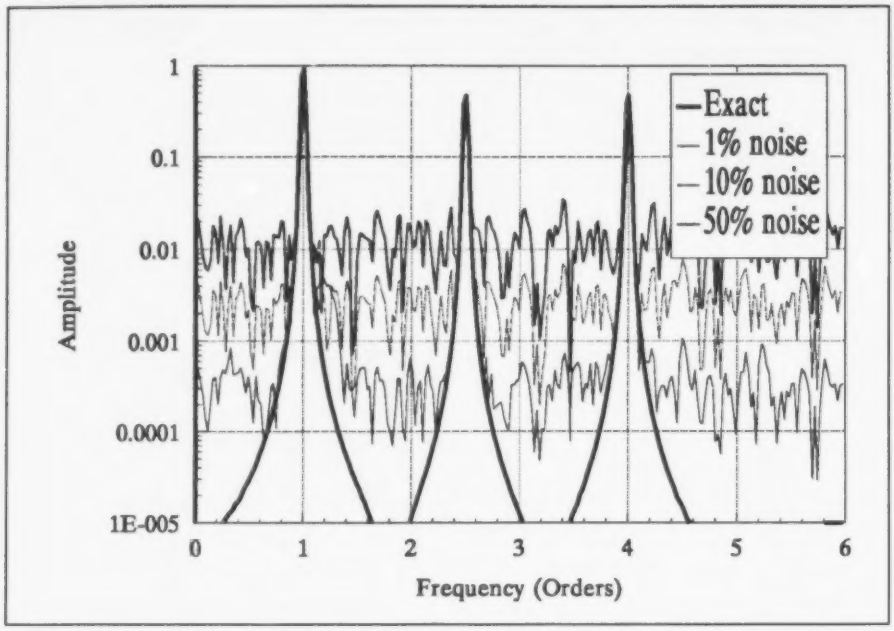
The approach taken to eliminate noise depends on its source. External noise sources are almost entirely dependent upon the specific circumstances of the vibration test and must be treated in the field by the user. Internal sources of noise are designed into the method and must be treated by the designer. This study suggests that, given enough processing power, computed order tracking could virtually eliminate internal noise.

4.4.5.2 Effects of Noise on Accuracy of the HP Method

Noise in the data manifests itself in the spectra as a uniform increase in the noise floor. Figure 38 shows a spectral comparison between methods with various amounts of random internal noise added to the vibration signal. Figure 37 shows a spectral comparison where different analog-to-digital converters (ADC) were modeled to introduce external noise. The effects are identical in form to Figure 38. Thus it is shown that once noise is introduced, it is difficult to tell by examination of the spectra where it originated.



► Figure 37: Effects of quantization and digital word length



► Figure 38: Spectrum showing effects of random noise

4.4.5.3 Effects of Noise on Choice of Interpolation Method

Starting with a noisy raw signal, each interpolation method was used for resampling. No method appeared to lower the noise floor better than another. The higher order interpolations produced spectra with lower noise at high shaft speeds, but this also happened when the signal contained no noise (see Figures 30, 32, 34, 36, and Table 3), making it difficult to draw a definitive conclusion.

4.4.6 Effects of Data Block Size

The vast majority of FFT routines require specific quantities of input data points; usually, these quantities are results of 2^N (i.e., 256, 512, 1024,...). Highly efficient FFT routines can be written for data block sizes which have this property because of inherent symmetries in the FFT method. The number of data points used depends on the results desired. Increasing the number of data points results in higher frequency resolution; it also increases calculation time. Although a high frequency resolution is desirable, a long calculation time is unacceptable for real-time implementation of the method.

In this study, a data block size of 512 was used to reach a compromise between storage requirements and processing time on one hand and frequency resolution on the other.

When using order tracking analysis, another consideration comes into play. Sampling for order tracking requires that a fixed number of data points be taken per revolution (this study used 12 resample points per revolution). Thus, for a larger data block, more shaft revolutions are required. In a runup, the machine speed increases during the sampling process. If a large data block size is chosen, the method will have to wait through more shaft revolutions, and the data at the end of the block may be taken at a considerably higher speed than the first data points. The resulting spectrum may not be truly representative of the vibrations at the trigger speed. For example, a spectrum labeled 100 rpm but containing data sampled from 100 rpm to 250 rpm may not be an accurate representation of the vibration frequencies and amplitudes at 100 rpm.

It is also possible that the next trigger speed may be passed before enough data have been acquired for the last one. This effect has already been discussed in Section 4.3.3.2, Misrepresentation of Frequency Content.

5 Summary & Conclusions

Computed order tracking has recently been introduced to aid in the vibration analysis of rotating machinery. This procedure requires simpler and lower cost equipment than that associated with traditional analog equipment. This new computational approach has not been previously examined to determine which inherent factors and assumptions have the greatest effects on accuracy. This study was executed to examine these issues. To carry this out, a digital simulation was performed that includes modeling the rotating machine, transducers, hardware and processing algorithm. Use of a total simulation prevented any external effects from influencing the tests, as would be expected with a physical apparatus. In addition, this simulation was able to determine the exact results, which were used as a base for comparison.

To test the effects of various factors, a control case was run, then one variable was changed and another test was run. The spectral results from these tests were compared using a newly developed *similarity factor*, which could range from zero (completely dissimilar) to one (identical).

Most problems with spectral accuracy are directly attributable to techniques incorporated within the method. Thus, with better software (and perhaps more computational power) COT can become increasingly more accurate.

In all tests performed, use of higher sampling rates on keyphasor and data signals resulted in improved accuracy. With raw data points closer together, the amplitude of the interpolated data will be more accurate. Recognizing that an excessive sample rate wastes computer resources by storing large quantities of raw data, it is important to note that increases in the sample rate above the Nyquist limit improve similarity. The best sample rate strikes a balance between data storage demands and accuracy.

The results presented in this thesis showed that the single greatest increase in spectral accuracy resulted from the improvements in precision of measuring the keyphasor pulse arrival time. If a data acquisition system is used which requires that both keyphasor and transducer channels use the same sample rate, then a sampling rate which produces good keyphasor timing accuracy should be chosen, as long as it is above the Nyquist limit for the transducer signal. This may require that much more transducer data be taken than is needed to ensure good keyphasor pulse timings.

Use of higher order interpolation also improved accuracy. Implementing a cubic spline interpolation instead of a linear interpolation reduced background noise by approximately on order of magnitude and improved similarity by

250% from 0.16 to 0.56. Potter didn't include details of his interpolation filter, but suggests that amplitude accuracy of 0.08% is theoretically attainable using linear interpolation filtering and 2x oversampling. These accuracy levels are comparable to this study's results from cubic spline interpolation.

Results of using interpolated digital filtering show it to compare favorably to fixed-coefficient digital filtering, especially during extreme rates of rotation acceleration. Caution must be used, however, since the filter interpolation depends on the accuracy of timing the keyphasor pulse arrival times.

6 Recommendations for Future Development

Having developed a flexible testing platform for the completed COT analysis, a similar study could be carried out on the B&K method with the purpose of comparing it to both the classical method and to HP's method.

Potter describes the use of an *interpolation filter* to perform COT. It would be interesting to learn about this type of filtering and incorporate the interpolation filter in the simulation to examine its strengths and weaknesses.

Digital filtering was added to this study to complete the simulation, but particular attention was not paid to designing a high performance filter. Determination of the minimum or optimal digital filter requirements for COT would provide valuable information.

For the simulation, more attention was paid to faithful modeling of the methods than to computational speed or efficiency. To move from testing by simulation to testing actual machines would require implementing the findings of this study and optimizing the computational techniques. If fast computing hardware is available, a real-time implementation could be developed and tested.

7 Bibliography

[1] Bellanger, Maurice G. Adaptive Digital Filters and Signal Analysis. New York, Marcel Dekker, 1987.

[2] Bentley Book One, Bentley Nevada Corp.

[3] Berry, J. "How to Track Rolling Element Bearing Health with Vibration Signature Analysis", Sound and Vibration. (Nov 1991); 24-35.

[4] Bogner, R.E., et. al., eds. Introduction to Digital Filtering. London, John Wiley & Sons, 1975.

[5] Hamming, R.W. Digital Filters. 3rd ed. Englewood Cliffs, New Jersey, Prentice Hall, 1989.

[6] Harris, Cyril M., et. al., eds. Shock and Vibration Handbook. 3rd ed. New York, McGraw-Hill, 1988.

[7] Hewlett-Packard Application Note 243-1, Dynamic Signal Analyzer Applications. Hewlett-Packard Co.

[8] Jackson, Charles. The Practical Vibration Primer. Houston, Gulf Publishing, 1979.

[9] Lipovszky, Gyrgy, and Kroly Slyomvri, and Gbor Varga. Vibration Testing of Machines and Their Maintenance. trans. S. Bars. Studies in Mechanical Engineering 10. Amsterdam, Elsevier Science, 1990.

[10] Mathew, J. and A. Szczepanik. "Monitoring the Vibrations of Low and Variable Speed Gears. " Diagnostics, Vehicle Dynamics and Special Topics. ed. T.S. Sankar. New York, ASME, 1989.

[11] Pope, Joseph. Basic Studies of Automobile Tire Noise. Ph.D. dissertation. Stanford U, 1978.

[12] Potter, Ronald W. Tracking and Resampling Method and Apparatus for Monitoring the Performance of Rotating Machines. United States Patent #4,912,661.

[13] Potter, Ron. "A New Order Tracking Method for Rotating Machinery", Sound and Vibration. (Sept. 1990): 30-34.

[14] Potter, Ron, and Mike Gribler. "Computed Order Tracking Obsoletes Older Methods." SAE Noise and Vibration Conference. (May 16-18, 1989): 63-67.

[15] Rogers, Robert J., and Richard V. Murphy. Machinery Condition Monitoring: Report for the Canadian Electrical Association. Fredericton, University of New Brunswick, 1980.

[16] Tan, C.N. and J. Mathew. "Monitoring the Vibrations of Variable and Varying Speed Gearboxes." The Institution of Engineers Australia, Vibration and Noise Conference, Melbourne 18-20 Spetember 1990.

[17] Atkin, Oktay. PC-DSP Ver 1.2. Englewood Cliffs, NJ. Prentice-Hall. 1990.

[18] Press, W. H. et. al. Numerical Recipes in C. University of Cambridge Press. 1989.

[19] Chapra, S. C. and Canale, R.P. Numerical Methods for Engineers. 2d. Ed. New York. McGraw-Hill.

[20] Herlufsen, H. "Bruel & Kjaer Sales Training Note: Digital Tracking using the 3550." Denmark. Bruel & Kjaer.1993.

[21] Brigham, E. O. The Fast Fourier Transform. Englewood Cliffs, NJ. Prentice-Hall. 1974.

8 Appendices

8.1 Fundamental code

8.1.1 Main program

```
// begin main program
int main (int argc, char *argv[])
{
    // declare pointers and other variables
    CMPLX FFTData[FFTMAX]; /* ptr to array of fft input data */
    DD DeltaT; /* used for Classical method */
    DD Jitter; /* used for Classical method */
    DD Rpm; /* rotational speed */
    DD t1=(DD)0;
    DD t2=(DD)0;
    DD RpmTrig=(DD)0; /* speed to start taking data */
    int i=(int)0; /* counter */

    /*
    ** clear the screen
    ** open data files
    ** get time now for run ID number
    ** initialize variables
    ** can make this interactive later
    */
    _clearscreen(_GCLEARSCREEN);
    _displaycursor(_GCURSOROFF);

    if((data_fp=fopen(OUT1,"wt"))==NULL) crash("File problems");
    file_write(99,(DD)0,(DD)0,(DD)0);
    if((fftnew_fp=fopen(OUT2,"wt"))==NULL) crash("File problems");

    ini(INI);

    // initialize variables
    KPDiff=(DD)2/(KPSampleRate);

    if(IfPrn) {
        _settextposition(19,0);
        if(ResampleMethod==COMPUTED_ORDER_TRACKING) {
            if(FilteringUsed)
                printf("Filtering is in use.\n");
            else
                printf("Filtering is not in use.\n");
        }
    }

    // do until runup is complete
    for(RpmTrig=RpmInit;RpmTrig<=RpmMax;RpmTrig += DeltaRpm) {
        if(IfPrn) {
            _settextposition(1,0);
            printf("Starting a main loop \n");
        }

        // reset variables
        for(i=0;i<FFTMAX;i++) FFTData[i].real=FFTData[i].imag=(DD)0;

        // choose the method of obtaining data for spectral analysis
        switch(ResampleMethod) {
            case COMPUTED_ORDER_TRACKING:
                if(FilteringUsed) LoadFilters(FLT);
                t1=keyphasor_signal(TimeNow);
                do {
                    do {
                        increment_time();
```

```
            t2=keyphasor_signal(TimeNow);
        }
        while((t2-t1)<KPDiff);
        Rpm=determine_rotational_speed(t2-t1,KeyPhasorSep,rpm);
        if(IfPrn) {
            _settextposition(2,0);
            printf("rpm = %5.1lf\tTime= %5.1lf\n",Rpm,TimeNow);
        }
        t1=t2;
    } while(Rpm<RpmTrig);

    HP_method(TimeNow,FFTData,BlockSize);

    break;
case EXACT_ORDER_TRACKING:
    t1=keyphasor_signal(TimeNow);
    do {
        do {
            increment_time();
            t2=keyphasor_signal(TimeNow);
        }
        while((t2-t1)<KPDiff);
        Rpm=determine_rotational_speed(t2-t1,KeyPhasorSep,rpm);
        if(IfPrn) {
            _settextposition(2,0);
            printf("rpm = %5.1lf\tTime= %5.1lf\n",Rpm,TimeNow);
        }
        t1=t2;
    } while(Rpm<RpmTrig);

    exact_method(FFTData,RpmTrig);

    break;
case CLASSICAL_ORDER_TRACKING:
    i=0;
    DeltaT=(DD)1/SampleRate;
    t1=keyphasor_signal(TimeNow);
    do {
        do {
            // increment_time
            TimeNow+=DeltaT;
            t2=keyphasor_signal(TimeNow);
        }
        while((t2-t1)<KPDiff);
        Rpm=determine_rotational_speed(t2-t1,KeyPhasorSep,rpm);
        if(IfPrn) {
            _settextposition(2,0);
            printf("rpm = %5.1lf\tTime= %5.1lf\n",Rpm,TimeNow);
        }
        t1=t2;
    } while(Rpm<=RpmTrig);

    // at trigger speed
    // print trigger speed to file
    fprintf(fftnew_fp,"%17.12lf",(DD)Rpm);
    do {
        // set sample rate
        DeltaT=(DD)60/(Rpm*(DD)SamplesPerRev);
        SampleRate=(DD)1/DeltaT;

        do {
            // increment time
            TimeNow+=DeltaT;

            // take a sample
            FFTData[i].real=transducer_signal(TimeNow,TRUE);
            i+=1;
            if(i>=BlockSize) break;

            // new keyphasor pulse arrive?
```

```
                t2=keyphasor_signal(TimeNow);
            }
            while((t2-t1)<KPDiff);

            // got new KP pulse, determine sample rate
            if(i<BlockSize) {
                Rpm=determine_rotational_speed(t2-t1,KeyPhasorSep,rpm);

                // module to generate error in Rpm measurement
                do { Jitter=rand(); } while ((Jitter<0)||(Jitter>RAND_MAX));
                Jitter=(DD)JitterAmpl*(((DD)Jitter/(DD)RAND_MAX)-(DD)0.5);

                // add error to Rpm
                Rpm+=Jitter*Rpm;

                if(IfPrn) {
                    settextposition(2,0);
                    printf("rpm=%5.1lf\tTime=%5.1lf\tRate=%5.1lf\n",-
                        Rpm,TimeNow,SampleRate);
                }
            }

            t1=t2;
        }
        while(i<BlockSize);
        break;
    case BORING_CASCADE_PLOT:
        t1=keyphasor_signal(TimeNow);
        do {
            do {
                increment_time();
                t2=keyphasor_signal(TimeNow);
            }
            while((t2-t1)<KPDiff);
            Rpm=determine_rotational_speed(t2-t1,KeyPhasorSep,rpm);
            if(IfPrn) {
                _settextposition(2,0);
                printf("rpm = %5.1lf\tTime= %5.1lf\n",Rpm,TimeNow);
            }
            t1=t2;
        } while(Rpm<=RpmTrig);

        for(i=0;i<BlockSize;i++) {
            FFTData[i].real=transducer_signal(TimeNow,TRUE);
            increment_time();
        }
        break;
    default:
        crash("Invalid analysis method in main().");
        break;
}

/*
** apply a window to the data
** save sample data
** call FFT routine - results returned in FFTData
** scale FFTData
** save FFT results
** convert FFT data into the desired format (magnitude,real&imag,etc)
** display FFT results
** set next desired rotational speed
*/
if(IfPrn) {
    _settextposition(1,0);
    printf("Finished sampling a complete block.\n");
    _settextposition(1,0);
    printf("Applying window to sample data. \n");
}

if(apply_window(Window,FFTData,BlockSize)!=0)
```

```
            crash("Problems in apply_window().");

        if(IfPrn) {
            _settextposition(1,0);
            printf("Performing FFT \n");
        }

        fft((((DD *)FFTData)-1),(TWO_BYTE)BlockSize,(TWO_BYTE)1);

        for(i=0;i<BlockSize;i++) {
            FFTData[i].real=(DD)2*FFTData[i].real/(DD)BlockSize;
            FFTData[i].imag=(DD)2*FFTData[i].imag/(DD)BlockSize;
        }

        if(IfPrn) {
            _settextposition(1,0);
            printf("Converting FFT results \n");
        }

        convert_fft_results(FFTData,BlockSize,DisplayFormat);

        if(IfPrn) {
            _settextposition(1,0);
            printf("Saving results to a file. \n");
        }

        if(FirstTime) {
            // fprintf(fftnew_fp,"%17.12lf",(DD)0);
            for(i=0;i<(BlockSize/2);i++)
                if(ResampleMethod==BORING_CASCADE_PLOT)
                    printf(fftnew_fp,",%17.12lf",((DD)i*(DD)SampleRate/(DD)BlockSize));
                else
                    printf(fftnew_fp,",%17.12lf",((DD)i*~
                        (DD)SamplesPerRev/(DD)BlockSize));
            fprintf(fftnew_fp,"\n");
        }

        if(ResampleMethod!=CLASSICAL_ORDER_TRACKING)
            fprintf(fftnew_fp,"%17.12lf",(DD)Rpm);
        for(i=0;i<(BlockSize/2);i++)
            fprintf(fftnew_fp,",%17.12lf",(DD)2*(DD)FFTData[i].real);
        fprintf(fftnew_fp,"\n");

        if(!IfPrn) {
            if(display_results(FFTData,BlockSize,FirstTime,MAGNITUDE)!=(TWO_BYTE)0)
                crash("Couldn't plot.");
        }
        else {
            _settextposition(1,0);
            printf("About to start another main loop \n");
        }

        FirstTime=FALSE;
    }

    /*
    ** close all output files
    ** return screen to normal
    ** end main program for computed order tracking
    */
    if(IfPrn) {
        _settextposition(1,0);
        printf("Freeing memory and closing files. \n");
    }
    fcloseall();
    fflush(stdin);
    _setvideomode(_DEFAULTMODE);
    return 0;
}
```

8.1.2 Hewlett Packard Method of Computed Order Tracking

```
int HP_method (DD t1, CMPLX *fft_data, int blocksize)
{
    extern BOOL FilteringUsed;
    extern BOOL IfPrn;
    extern DD KeyPhasorSep;
    extern DD KPDiff;
    extern DD TimeNow;
    extern int SamplesPerRev;

    DD KP[KPMAX];
    DD t2=(DD)0;
    DD t3=(DD)0;

    int i=(int)0;
    int i1=(int)0;
    int i2=(int)0;
    int i3=(int)0;
    int j=(int)0;
    int kp_pulses_needed=(int)0;
    int n=(int)0;
    int points_stored=(int)0;

    struct SAMPLE raw[RAWMAX];
    // struct SAMPLE *raw;
    struct SAMPLE resampled[RSMPLSIZE];

    // intialize variables
    i=j=0;
    KP[0]=keyphasor_signal(TimeNow); /* KP[0]=t1; doesn't work */
    kp_pulses_needed=(int)ceil(((DD)blocksize*(DD)2*(DD)PI)/((DD)SamplesPerRev*-
        (DD)KeyPhasorSep))+(int)4;
    if(kp_pulses_needed>KPMAX)
        crash("Too many keyphasor pulses needed in HP_method().");
    for(i=0;i<RAWMAX;i++) {
        raw[i].data=raw[i].time=(DD)0;
        if(i<RSMPLSIZE) resampled[i].data=resampled[i].time=(DD)0;
    }

    // record start of block and KP pulse
    file_write(4,TimeNow-0.0000001,0,0);
    file_write(2,t1,0,0);

    // acquire raw data
    for(i=1;i<kp_pulses_needed;i++) {
        KP[i]=KP[i-1];
        // record KP pulse
        file_write(2,KP[i-1],0,0);
        while(KP[i]-KP[i-1]<KPDiff) {
            raw[j].data=transducer_signal(TimeNow,TRUE);
            raw[j].time=TimeNow;
            j+=1;
            if(j>=RAWMAX)
                crash("Not enough room in raw[] in HP_method().");
            KP[i]=keyphasor_signal(TimeNow);
            increment_time();
        }
    }

    // filter raw data
    if(FilteringUsed) FilterRawData(raw,KP,j,i);

    // begin resampling
    t1=KP[1];
    t2=KP[2];
    i1=0;
    i2=0;
    j=0;
```

```
    // find the raw[] index that corresponds to the second KP pulse
    while(i2==0) {
        if(raw[j].time>=t2) i2=j;
        j+=1;
        if(j>RAWMAX) crash("Not enough room in raw[] in HP_method().");
    }
    i3=i2;

    j=0;
    for(i=3;i<kp_pulses_needed;i++) {
        t3=KP[i];

        // find the raw[] index that corresponds to the next KP pulse
        while(i3==i2) {
            if(raw[j].time>=t3) i3=j;
            j+=1;
            if(j>RAWMAX) crash("Not enough room in raw[] in HP_method().");
        }

        // resample the data within three contiguous data points
        n=resample_data(t1,t2,t3,raw+i1,resampled,i3-i1+1,SamplesPerRev);

        // copy resampled data points into finished data block
        for(j=0;j<n;j++) {
            fft_data[points_stored].real=resampled[j].data;
            // record resampled data point
            file_write(3,resampled[j].time,resampled[j].data,0);
            if(IfPrn) {
                _settextposition(18,0);
                printf("Data block percent full: %5.1lf\n"-
                    ,(DD)points_stored/(DD)blocksize*(DD)100);
            }
            points_stored += 1;
            if(points_stored>=blocksize) goto End;
        }

        // shift data to next keyphasor pulse in the sequence
        t1=t2;
        t2=t3;
        i1=i2;
        i2=i3;
    }

End:
    // record end of data block
    file_write(5,TimeNow+0.0000001,0,0);
    // free(raw);
    return 0;
}
```

8.1.3 Subroutine for Resampling Data

```
int resample_data(DD t1,DD t2,DD t3,struct SAMPLE *raw,struct SAMPLE *rsmpl,int nraw,int
npr)
{
    /*
    ** input:
    ** t1,t2,t3: arrival times of three contiguous KP pulses
    ** raw: transducer output, sampled at constant delta_t
    ** rsmpl: resampled data
    ** nraw: number of points in raw[]
    ** npr: desired number of resamples per shaft revolution
    **
    ** output:
    ** none
    */
    extern int InterpMethod;
```

```
    extern DD KeyPhasorSep;
    DD *X,*Y;
    DD *b,*c,*d;
    int i,n;

    // determine resample times
    n=ResampleTimes(t1,t2,t3,npr,KeyPhasorSep,rsmpl);

    // resample the data
    switch(InterpMethod) {
        case LINEAR:
            LinearInterp(raw,rsmpl,nraw,n);
            break;
        case QUADRATIC:
            QuadraticInterp(raw,rsmpl,nraw,n);
            break;
        case PIECECUBIC:
            PieceWiseCubic(raw,rsmpl,nraw,n);
            break;
        case BLOCKCUBIC:
            if((X=(DD *)calloc(nraw,sizeof(DD)))==NULL)
                crash("Couldn't allocate memory for X[] in resample_data().");
            if((Y=(DD *)calloc(nraw,sizeof(DD)))==NULL)
                crash("Couldn't allocate memory for Y[] in resample_data().");
            if((b=(DD *)calloc(nraw,sizeof(DD)))==NULL)
                crash("Couldn't allocate memory for b[] in resample_data().");
            if((c=(DD *)calloc(nraw,sizeof(DD)))==NULL)
                crash("Couldn't allocate memory for c[] in resample_data().");
            if((d=(DD *)calloc(nraw,sizeof(DD)))==NULL)
                crash("Couldn't allocate memory for d[] in resample_data().");

            for(i=0;i<nraw;i++) {
                X[i]=raw[i].time;
                Y[i]=raw[i].data;
            }

            Spline(X,Y,nraw,b,c,d);

            for(i=0;i<n;i++)
                rsmpl[i].data=SplineEval(X,Y,nraw,b,c,d,rsmpl[i].time);

            free(X);
            free(Y);
            free(b);
            free(c);
            free(d);

            break;
        default:
            crash("Illegal resample method in resample_data().");
            break;
        }

    // end function resample data
    return n;
}
```

8.1.4 Subroutine for Determining Resample Times

```
int ResampleTimes(DD t1,DD t2,DD t3,int npr,DD kps,struct SAMPLE rsmpl[])
{
    /*
    ** input:
    ** t1,t2,t3: arrival times of three contiguous KP pulses
    ** npr: desired number of resample times per revolution
    ** kps: keyphasor separation in radians (angle through which
    ** shaft will rotate between keyphasor pulses)
```

```
 **
 ** output:
 ** rsmpl[]: vector containing resample times
 ** return val: number of resample times
 */

 int j,k,m,n;
 int k_min,k_max;

 DD arg;
 DD b0,b1,b2;
 DD det;
 DD delta_theta;
 DD tmp1,tmp2,tmp3;

 delta_theta=((DD)2*(DD)PI)/(DD)npr;

 /* calculate determinant
 ** exit if determinant is zero
 ** calculate coefficients
 */
 if((det=(t2-t1)*(t3-t1)*(t3-t2))==(DD)0)
     crash("Determinant z    in resample_data().");
 tmp1=kps/det;
 b0=tmp1*t1*(t3*(t1-t3)+((DD)2*t2*(t2-t1)));
 b1=tmp1*((t1*t1)-((DD)2*t2*t2)+(t3*t3));
 b2=tmp1*(-t1+(2*t2)-t3);

 // calculate k_min (must be greater than or equal to tmp1)
 if(delta_theta==(DD)0)
     crash("delta_theta zero in determine_resample_times");
 tmp1=kps/((DD)2*delta_theta);
 k_min=(int)ceil((DD)tmp1);

 /* calculate k_max (must be less than tmp1)
 ** if k_max is not greater than k_min, exit with error message
 */
 tmp1 *= (DD)3;
 k_max=(int)floor((DD)tmp1);
 if((DD)k_max==tmp1) k_max-=(int)1;
 if(k_max<=k_min)
     crash("k_max not greater than k_min in determine_resample_times");

 /* determine number of resample times
 ** zero the elements of rsmpl[]
 */
 n=(int)(k_max-k_min+(int)1);
 for(m=0;m<n;m++) {
     rsmpl[m].data=(DD)0;
     rsmpl[m].time=(DD)0;
 }

 // calculate the resample times
 if(b2!=(DD)0) {
     tmp1=(DD)1/((DD)2*b2);
     tmp2=(DD)4*b2;
     tmp3=b1*b1;

     for(k=k_min;k<=k_max;k++) {
         arg=(DD)sqrt((DD)(tmp2*(((DD)k*delta_theta)-b0)+tmp3));
         if(arg==(DD)0)
             crash("Negative square root argument in determine_resample_times()");
         j=k-k_min;
         rsmpl[j].time=tmp1*(arg-b1);
     }
 }
 else {
     for(k=k_min;k<=k_max;k++) {
         j=k-k_min;
         rsmpl[j].time=(((DD)k*delta_theta)-b0)/b1;
```

```
        }
    }
    return n;
}
```

8.2 Sample Initialization File

```
[resampling]
OrderTrackingMethod=Computed
InterpolationMethod=BlockCubic < computed method only >
SamplesPerRevolution=12 < samples needed to avoid aliasing >
FilteringUsed=No < lowpass filtering to eliminate high orders >
FilterInterpolation=No < improve(?) filters by adjusting for rpm >

[machine signal coefficients]
KeyphasorsPerRevolution=1 < number of keyphasor pulses per revolution >
A0=0 < radians >
A1=0 < radians/sec >
A2=1.0472 < radians/sec/sec >
A3=0 < radians/sec/sec >
A4=0
Amplitude=1 < base signal amplitude >
NoiseAmplitude=0.0 < random noise in transducer signal >

[fourier analysis]
BlockSize=512 < number of elements in data block >
Window=Hanning < window applied to data prior to FFT >

[startup]
StartTime=35 < simulation start time in seconds >

[sampling]
SampleRate=250 < Hz >
RpmOfFirstSpectra=400 < rpm at which sampling is first triggered >
RpmOfLastSpectra=1099 < no sampling triggered after this speed >
RpmBetweenSpectra=100 < rpm between sampling triggers >
KeyPhasorNoise=0.0 < random error in timing of kephasor signal >
KeyPhasorSampleRate=50000 < Hz >
JitterAmplitude=0.0 < random noise in rpm, classical method >

[a/d converter] < not in use yet >
UseADC=No < simulate prescence of ADC >
MaxVolts=5 < maximum volts readable by ADC >
MinVolts=-5 < minimum volts readable by ADC >
WordLength=8 < A/D converter output word size in bits >
Gain=1 < to fill range of ADC >

[output]
OutputFormat=Magnitude < output data format, magnitude of real and imag >
GraphicsEnabled=No < do or don't draw the spectra on-screen >
```